

# Faithful Visualisation of Similarities in High Dimensional Data



**Jiixin Kou**

**Supervisors: Chris Watkins, Zhiyuan Luo**

Computer Learning Research Centre

Department of Computer Science

Royal Holloway, University of London

This dissertation is submitted for the degree of  
*Doctor of Philosophy*

August 2016



I would like to dedicate this thesis to my loving mother.





# **Declaration**

## **Declaration of Authorship**

I hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Name: Jiaxin Kou

Signed:

Date:

Jiaxin Kou

Supervisors: Chris Watkins, Zhiyuan Luo

August 2016



## **Acknowledgements**

First and foremost I would like to thank my supervisor Chris Watkins and Zhiyuan Luo for their excellent supervision. Also I extend special thanks to Sue and Kevin for their kindness and support. I also thank my fellow colleague for being such good companies in the department, it is a great time I have spend in the UK. Finally my parents and my grandfather deserve my special thanks, my work is not possible without their forever love and caring.



# **Abstract**

In the last fifteen years, new methods of dimension reduction have been invented that enable much improved visualisation of high-dimensional data-sets. Conventionally, the data-sets are visualised as two-dimensional scatterplots, and similarity relationships between data-cases are revealed by grouping and proximity of points in the plane. But the arrangement of points in a 2D scatterplot cannot faithfully represent complex high-dimensional structure: more expressive 2D visualisations are needed.

This thesis develops new types of diagram that can represent data-similarities more expressively than a mere scatterplot. The approach is to automatically select a graph to overlay on the scatterplot, in order to enable a richer visualisation of similarities than is possible by the arrangement of points alone, and to correct distortions inherent in scatterplot visualisation.

Methods and software are developed for selecting and graphically representing the overlay graph as a diagram that humans can read. These diagrams enable correct and informative human interpretation of scatterplots that would otherwise be hard to interpret or misleading.



# Table of contents

<b>List of figures</b>	<b>xv</b>
<b>List of tables</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Scatterplot Visualisation of High Dimensional Data</b>	<b>5</b>
2.1 Overview . . . . .	5
2.2 Dataset Showcase . . . . .	8
2.3 Principal Component Analysis . . . . .	9
2.3.1 Overview . . . . .	9
2.3.2 Visualising Spherical Shell by PCA . . . . .	10
2.4 Beginnings of NLDR: MDS and Its Variants . . . . .	13
2.4.1 Classical MDS and Metric MDS . . . . .	13
2.4.2 Sammon Mapping and Curvilinear Component Analysis . . . . .	14
2.4.3 Visualising Spherical Shell by NLM . . . . .	16
2.5 Isomap . . . . .	17
2.5.1 Visualising Spherical Shell by Isomap . . . . .	17
2.6 Locally Linear Embedding . . . . .	18
2.6.1 Visualising Spherical Shell by LLE . . . . .	19
2.7 Stochastic Neighbour Embedding and its variants . . . . .	19
2.7.1 Affinity instead of Distance . . . . .	21
2.7.2 Perplexity as Soft Neighbourhood Selection . . . . .	22
2.7.3 Symmetric SNE . . . . .	23
2.7.4 Heavy Tail Distribution . . . . .	26
2.7.5 Cost Function and Optimisation . . . . .	26
2.7.6 Visualising Spherical Shell by t-SNE . . . . .	29
2.8 Conclusion . . . . .	29

<b>3</b>	<b>Scatterplot Visualisation Quality Assessment</b>	<b>31</b>
3.1	Overview . . . . .	31
3.2	Shepard Plot and Co-Ranking Histogram . . . . .	32
3.3	Venna's Trustworthy and Continuity . . . . .	33
3.3.1	Using Venna's T & C to Measure Visualisation of The 3D Sphere . . . . .	36
3.4	Venna's Precision and Recall . . . . .	37
3.4.1	Using Venna's P & R to Measure Visualisation of Spherical Shell . . . . .	39
3.5	CheckViz . . . . .	39
3.5.1	CheckViz of PCA Visualisation on 3D Sphere . . . . .	42
3.6	Neighbour Plot . . . . .	44
3.6.1	The Neighbour Plot of the 3D Sphere Visualisation . . . . .	45
3.7	Conclusion . . . . .	45
<b>4</b>	<b>Delaunay Plot</b>	<b>47</b>
4.1	Overlay Graph . . . . .	47
4.2	Motivation of Delaunay Plot . . . . .	48
4.3	Geometric Properties of Delaunay Triangulation . . . . .	48
4.4	Primitive Delaunay Plot . . . . .	49
4.5	Refined Delaunay Plot . . . . .	52
4.5.1	Cutting Long Delaunay edges . . . . .	52
4.5.2	Stochastic Neighbour . . . . .	53
4.5.3	New Visual Design of Delaunay Plot . . . . .	56
4.5.4	Input Neighbourhood Estimation Scheme . . . . .	57
4.6	The Myth of Output Neighbourhood . . . . .	61
4.6.1	Defining Output Space Neighbourhood by Delaunay Neighbours . . . . .	62
4.7	Objective Measures of Delaunay Plot . . . . .	63
4.8	The Implication of Transitive Proximity Relationship . . . . .	66
4.8.1	Delaunay Neighbour Reachable . . . . .	66
4.8.2	Noise in Delaunay Neighbour Reachable . . . . .	68
4.8.3	Faithful Delaunay Patches . . . . .	69
4.9	Examples of DP and FDPM . . . . .	71
4.10	Conclusion . . . . .	77
<b>5</b>	<b>Proxigram</b>	<b>79</b>
5.1	Overview . . . . .	79
5.2	From Neighbour Plot to Proxigram . . . . .	79
5.2.1	Definition . . . . .	79



5.2.2	Visual Appearance . . . . .	80
5.2.3	Visual Cues in General Proxigram . . . . .	81
5.2.4	Examples of General Proxigram . . . . .	82
5.3	Level of Truth . . . . .	84
5.4	Tear-up Proxigram . . . . .	87
5.4.1	Definition . . . . .	87
5.4.2	Tear-up Filtering Scheme . . . . .	87
5.4.3	Categories of Tear-up . . . . .	88
5.4.4	Proxi-Efficiency . . . . .	89
5.4.5	Bundling the Tear-up Proxi-Arrows . . . . .	89
5.4.6	Example of Tearup-Proxigram . . . . .	90
5.5	Spanner Graph from High Dimensional Data . . . . .	93
5.5.1	Overview . . . . .	93
5.5.2	Greedy Spanner Graph . . . . .	93
5.5.3	Short Conclusion . . . . .	95
5.6	Exploring Mutual Proximities in Proxigram . . . . .	96
5.6.1	Mutual Proximity Closure . . . . .	96
5.6.2	A Greedy Strategy of Searching Mutual Proximity Closures . . . . .	96
5.6.3	Minimum Spanning Forest of Mutual Proximity Closures . . . . .	97
5.6.4	Properties of Faithful Proximity Forest . . . . .	98
5.7	Conclusion . . . . .	102
<b>6</b>	<b>Tapestry Plot</b>	<b>103</b>
6.1	Control of Uncertainty . . . . .	103
6.2	Work-flow of Visualising High Dimensional Data . . . . .	106
6.3	Visualisation Sequences of Spherical Shell . . . . .	108
6.4	Visualisation Sequences of Earth Cities . . . . .	122
6.5	Visualisation Sequences of Snakes Dataset . . . . .	135
6.6	Visualisation Sequences of MNIST . . . . .	136
6.7	Visualisation Sequences of Glass Dataset . . . . .	144
<b>7</b>	<b>Conclusion</b>	<b>153</b>
7.1	What has been Achieved? . . . . .	153
7.2	Future Work . . . . .	154
	<b>References</b>	<b>155</b>

<b>Appendix A</b>	<b>Edge Bundling</b>	<b>159</b>
A.1	Background of Edge Bundling . . . . .	159
A.2	Edge Bundling Process . . . . .	160
A.3	Edge Bundling Rendering . . . . .	163
A.4	Edge Bundling Performance of Mingle . . . . .	163
A.5	FDEB Edge Compatibility Measures . . . . .	174
	A.5.1 Edge Similarity Modification . . . . .	176
	A.5.2 Ink Function Penalty Scheme . . . . .	176
A.6	Conclusion . . . . .	177

# List of figures

2.1	The 3D Spherical Shell . . . . .	8
2.2	Best Fit Line, The Intuition of PCA . . . . .	11
2.4	NLM Visualisation of 3D Sphere in pseudo-colours . . . . .	16
2.5	Isomap Visualisation of 3D Sphere in Pseudo-colours . . . . .	18
2.6	LLE Visualisation of 3D Sphere . . . . .	20
2.7	Two Gaussian clusters in 2D, overlay with affinity curves ( <i>perplexity</i> = 20). . . . .	23
2.8	Visualisation of data in Fig. 2.7 by t-SNE ( <i>perplexity</i> = 20) . . . . .	24
2.9	Visualisation of data in Fig. 2.7 by SNE ( <i>perplexity</i> = 40) . . . . .	25
2.10	The scatterplot in Fig. 2.8a; Contour Plot in Respect to Point <i>a</i> . . . . .	27
2.11	The Twin Clusters Example. . . . .	28
2.12	t-SNE Visualisation of 3D Sphere ( <i>perplexity</i> = 30) . . . . .	29
2.13	t-SNE Visualisation of the 3D Sphere in Different Perplexity. . . . .	30
3.1	Shepard Plot of the DR visualisation of the 3D Sphere . . . . .	34
3.2	Co-Ranking Histogram of the DR visualisation of the 3D Sphere . . . . .	35
3.3	Trustworthy and Continuity Performance for DR Methods to The 3D Sphere . . . . .	37
3.4	Mean-Precision against Recall for DR Methods to the 3D Sphere. (input neighbourhood size $r = 30$ ) . . . . .	40
3.5	CheckViz of PCA Visualisation of The 3D Sphere. . . . .	43
3.6	Neighbour Plots ( $k = 2$ ) of The 3D Sphere Visualisation. . . . .	46
4.1	The Solid Twin Cubes Dataset . . . . .	50
4.2	Primitive Delaunay Plot of MDS Visualisation to the Solid Twin Cubes. . . . .	50
4.3	Histogram of 2D $k$ -nearest neighbour selected in Fig. 4.2 . . . . .	52
4.4	LDECS-SN effects on Solid Twin Cubes Example . . . . .	55
4.5	Dense blue cube and sparse red cubs with one cyan outlier . . . . .	55
4.6	LDECS Effects on The Modified Cubes Example . . . . .	56
4.7	New Design of the Delaunay Plot Example in Fig. 4.6 . . . . .	57

4.8	Delaunay plot of the Modified Two Cubes Colour in INES-SN-Discrete (perplexity = 6) . . . . .	60
4.9	Histogram of INES-SN-Discrete Performance to Fig. 4.8 . . . . .	60
4.10	Delaunay plot of The Modified Cubes Coloured in INES-SN-Discrete (perplexity = 10) . . . . .	61
4.11	Histogram of INES-SN-Discrete Performance to Fig. 4.10 . . . . .	62
4.12	An Example of Missing a “nearby” point (green) for central point (red) in Delaunay plot . . . . .	65
4.13	Delaunay Plot of t-SNE Visualisation on The 3D Sphere . . . . .	72
4.14	Faithful Delaunay Patches Map of t-SNE Visualisation on The 3D Sphere . . . . .	73
4.15	3D Gaussian Spherical Shell Example . . . . .	74
4.16	Delaunay Plot of t-SNE Visualisation on Gaussian Sphere . . . . .	75
4.17	Faithful Delaunay Patches Map of t-SNE visualisation of 3D Gaussian Sphere . . . . .	76
5.1	General Proxigram of the 3D Gaussian Sphere, INES-KNN . . . . .	83
5.2	General Proxigram of the 3D Gaussian sphere, INES-SN . . . . .	84
5.3	The $\lambda$ -Proxigram ( $\lambda = 0.8$ ) of the 3D Gaussian sphere, INES-SN . . . . .	86
5.4	The tearup-Proxigram applied to Fig. 5.3, INES-SN vs DN(2) . . . . .	91
5.5	The bundled tearup-Proxigram of the t-SNE visualisation . . . . .	92
5.6	Spanner Example: 50 random points in 2D . . . . .	94
5.7	Spanner Example: 25 points Circle in 2D . . . . .	95
5.8	The Faithful Proximity Forest of the Gaussian Sphere, INES-kNN( $k = 30$ ) . . . . .	101
6.1	Global Qualities for t-SNE on Glass Data . . . . .	145
6.2	PCA Visualisation of Glass Dataset . . . . .	146
6.3	t-SNE (perplexity = 20) Visualisation of Glass Dataset . . . . .	147
6.4	Delaunay Plots of t-SNE Visualisation Coloured in HD Distance . . . . .	148
6.5	Delaunay Plot of t-SNE Visualisation Coloured in SN score (perplexity = 20) . . . . .	149
6.6	Proxigram ( $k = 5$ ) of t-SNE Visualisation Coloured in SN (perplexity = 20) . . . . .	150
6.7	The Tapestry Plot of t-SNE Visualisation . . . . .	152
A.1	Mingle Algorithm Proof of Concept: $numOfLevels = 2, numOfRecursion = 1, penalty = 2, k = 2$ . . . . .	164
A.2	Proof of Concept of Edge Bundling Rendering: Two blue straight-line edges are bundled into a three-segments spline, shown by black lines. The Green dashed splines are the straight-line rendering of the spline with $\lambda = 0.7$ , while the red dashed splines are the same but draw in Bézier curves. . . . .	166
A.3	eastwestcommute dataset with 130 edges unbundled. . . . .	167
A.4	Mingle with only agglomerative edge bundling, edges reduced from 130 to 22. . . . .	168

A.5	Mingle after one recursive, edges reduced from 130 to 6. . . . .	169
A.6	The same setting as Fig. A.5, but $\lambda = 0.7$ . The splines are rendered in straight-line. Setting $\lambda$ not to 1.0 is not good, we should avoid in later applications because even all edges are bundled the rendering actually not saving ink but waste more ink than unbundled. . . . .	170
A.7	The same setting as Fig. A.5 but $p = 5$ , edges reduced from 130 to 2. Larger $p$ will favour bundling than small turning angle. Thus the result has more edges bundled than before, however the appearance of large turning angles are not only unaesthetic but also useless, because such aggressive bundling does the opposite of revealing high level edge patterns. . . . .	171
A.8	A showcase of the specific control of level of recursive and agglomerative bundling in Mingle Python implementation. Recursive times is set to 5 and Agglomerative bundling times to 1. Edges are reduced from 130 to 3. This will result the bundling have more segments than only 3, which can help the splines look more natural. In some cases specific control of coarsening levels can help avoid aggressive bundling during aggressive bundling process. . . . .	172
A.9	The same settings as Fig. A.8, but rendering in Bézier curves. It looks aesthetic and it is easy to follow where each edge is going. However like Fig. A.6, it does waste more ink to draw than the unbundled edges. . . . .	173



# List of tables

2.1	Milestone Dimensionality Reduction Methods . . . . .	<a href="#">7</a>
6.1	Glass Identification Class Information . . . . .	<a href="#">144</a>
6.2	Settings of Tear-up Proxigram . . . . .	<a href="#">151</a>
A.1	Default Mingle Parameter Settings . . . . .	<a href="#">163</a>





# Chapter 1

## Introduction

Visualisation is an important tool for humans to understand data. Today fields like data analysis and machine learning process high dimensional data extensively, so it is a natural interest to understand high dimensional data using visualisation. A typical high dimensional dataset (HD data) is a collection of  $n$  data items (observations), each with  $p$  numeric attributes (features), which can be represented as a matrix  $X$  with  $n$  rows and  $p$  columns.

One possible visualisation of the HD data is a **2D scatterplot** that resembles a meaningful data map, in which similar data items in HD space are placed closely in the 2D space while dissimilar ones remain far apart. The similarity between any two items is usually defined by Euclidean distance metric and there are  $n(n-1)/2$  such pairwise similarities. For two-dimensional data ( $p = 2$ ), a scatterplot shows the dataset and all  $n(n-1)/2$  similarities exactly. However for higher dimensional data, it requires a procedure of projecting the data items from HD space into 2D space, resulting a coordination matrix  $Y_{(n \times 2)}$  so that the proximities among data items in the scatterplot reflect their similarities in the maximum capacity.

At present, such projections (or embedding) are formalised as a special case of a field of study called dimensionality reduction (DR), by restricting its target dimension into two. Despite breakthroughs in recent decades, a 2D scatterplot can never represent high dimensional data faithfully if the intrinsic dimension of the data is more than two. For example, for a regular tetrahedron in 3D space, all six pairwise Euclidean distances (HD similarities) among these four points are mutually equal; a special orthogonal projection centred at one vertex can preserve three out of six edges to be equal, but the other three edges are shorter than their original. For a larger dataset with higher dimensionality, this fundamental limitation can introduce massive unexpected distortions in the 2D scatterplot, regardless of which DR method is used and how they could be developed in the future.

In general there are two types of distance distortion in this visualisation. Two points can be close in HD space but far apart in the visualisation, we call this a “tear-up” error. On the

other hand, two points can be relatively far apart in HD space, but close in the visualisation; we call this “false-neighbour” error. Defining “tear-up” and “false-neighbour” in a precise way is not simple, because we will need to determine suitable neighbourhood thresholds in both HD space and 2D space.

Furthermore this is not the only obstacle. The *curse of dimensionality* is a broad term for the difficulties that high dimensional data brings for statistical analysis. In particular, though the 2D data map does provide an intuitive view of the original data distribution, the geometric properties of high dimensional space are counter-intuitive. All our intuitions are based on the experience we learn from 2D or 3D space but they are only special cases when the dimensionality of space is less than four. For instance, with the dimensionality increasing, the distribution of all pairwise distances become concentrated at their average value due to the Law of Large Numbers. As a consequence the difference between the distance of the nearest neighbour to the furthest neighbour becomes small.

Due to the above challenges, the 2D scatterplot visualisation of complex HD data can be in fact not only obscure but also highly misleading. Thus the usability of the visualisation is not reliable. However I argue that visualising high dimensional data as a meaningful 2D scatterplot should remain straightforward as much as possible, because its objective is to serve as a natural observation to the data, not another puzzle. Therefore the fundamental difficulties suggest that HD data visualisation is a much greater task than dimensionality reduction alone can achieve. For that reason, I believe some essential visual annotations, design choices of the scatterplot and the quality metrics related to it are missing, yet very few studies in this problems have been done, save for [Lespinats and Aupetit \(2011\)](#).

The central problems I considered in this thesis are:

- What is the alternative appearance of the scatterplot of HD data, if it is seriously distorted?
- How is this visualisation related to human visual perception?
- What are the compromises between the complexity and readability of such visualisation?

To fill this research gap, the main contribution of this thesis is to propose the concept of “Overlay Graph”, a family of systematic augmentations to the 2D scatterplot visualisation of the HD data. The vital information is missed in the traditional HD data visualisation is compensated by the overlay graph, such that the 2D data map is more trustworthy and less ambiguous. Remarkably, I have not been able to find such “overlay graphs” in the literature.

Throughout this thesis, I used the state-of-art DR method t-SNE [Van der Maaten and Hinton \(2008\)](#) to construct the initial 2D layout of the HD data. But I regarded this algorithm only as a tool, and I did not improve it. My contribution is to devise a range of overlay graphs that

augments this t-SNE scatterplot. I will show that without these overlay graphs, the scatterplot visualisations produced by t-SNE or any other DR methods can be deceiving and lead to wrong interpretations of the data.

The structure of this thesis is as follows:

- Chapter 2 reviews some of the milestone dimensionality reduction method used for producing high dimensional data visualisation.
- Chapter 3 reviews some of the traditional and recent developed quality measures related to high dimensional data visualisation.
- Chapter 4 introduces the “Delaunay Plot”, the first overlay graph using Delaunay triangulation. It shows the HD similarities among nearby data points in the scatterplot and reveals false-neighbours. Later in the chapter we introduce the “Delaunay Neighbours”, an intuitive neighbourhoods definition in 2D space, helped by the Delaunay plot.
- Chapter 5 introduces the “Proxigram”, the second overlay graph in the format of an arrow plot. It aims to show true neighbours of all points in high dimensional space. The chapter proceeds with varieties of refinements for the proxigram, with each of the special kinds being more expressive and less cluttered.
- Chapter 6 introduces the final overlay graph, the “Tapestry Plot”, which is a more informative visualisation combining both the Delaunay plot and the proxigram. This overlay graph reveals both tear-up and false-neighbour errors in the scatterplot visualisation. With these distortions presented directly in the visualisation, the traditional quality measures become trivial, while the readability becomes a more important consideration for human to retrieve the similarity information. The chapter ends with experiments on several synthetic and real-world datasets.
- Chapter 7, I summarise achievements and outline several possible directions for future research.
- Finally, Appendix I includes edge bundling, an important design choice for the proxi-arrows in the proxigram and the tapestry plot. The topics covered in the appendixes will be briefly mentioned in several main chapters if related.



## Chapter 2

# Scatterplot Visualisation of High Dimensional Data

### 2.1 Overview

Dimensionality Reduction (DR) is the theoretical foundation of the traditional high dimensional data visualisation. It is commonly defined as the problem of finding a mapping from an input space (HD space)  $X \in \mathbb{R}^p$  of  $n$  vectors into an output space  $Y \in \mathbb{R}^{p^*}$  with  $p \gg p^*$ . The field has been studied for a long time and its first classical and perhaps the most important technique Principal Component Analysis was initially proposed more than a century ago by [Pearson \(1901\)](#). Since then, DR has evolved with the changing of specific research interests and applicable situations.

Although 2D scatterplot visualisation is one of the applications and research interests of DR, it must be noted that not all DR methods are specially developed for this type of problem. DR methods also contribute to other sub-fields such as feature extraction, feature selection, compression. In this chapter, I will only discuss some milestones that are widely used for high dimensional data visualisation. Also note that in this thesis, I restrict the scatterplot visualisation to two dimensions, not because one might feel that it makes little difference whether the data is reduced from  $N$  dimensionality to two or three (actually it makes lots of difference), but for accessibility. Humans are 3D creatures, so information is usually presented on the 2D plane.

Broadly speaking, DR techniques can be divided into early linear methods and more sophisticated non-linear methods known as NLDR. [van der Maaten et al. \(2009\)](#) gave an excellent comparative review on DR methods for visualisation and categorised these methods into those preserving global properties and local properties. Also as [Lee et al. \(2014\)](#) pointed

out the characteristics of DR methods were transformed from preserving data variance to pairwise distance, and then into similarity (modelled as probabilities) recently. Table. [2.1](#) shows all milestone of DR methods that are widely used for visualisation in chronological order with their “kinship” relationship, characteristics and optimisation properties.

Method	Reference	Kinship	Focus Prospect	Characteristic	Linear	Optimisation
PCA	Hotelling (1933)	-	Global Properties	Data Variance	Yes	Closed solution, Eigenvalue Problem
Classic MDS	Torgerson (1952)	-	Global Properties	Dot Product	Yes	Closed solution, Eigenvalue Problem
Sammon Mapping	Sammon (1969)	MDS	Global Properties	Distances	No	Minimise the sum of distance variance
CCA	Denartines and Hérault (1997)	Sammon Mapping	Global Properties	Distances	No	Minimise the sum of distance variance
Kernel PCA	Schölkopf et al. (1998)	PCA	Global Properties	Data Variance and Dot Product	No	Closed solution, Eigenvalue Problem
Isomap	Tenenbaum et al. (2000)	MDS	Global Properties	K-Nearest Neighbour Graph and Distance	No	Closed solution, Eigenvalue Problem
LLE	Roweis and Saul (2000)	-	Local Structure	K-Nearest Neighbour Graph and Distance	No	Closed solution, Eigenvalue Problem
Laplacian Eigenmap	Belkin and Niyogi (2003)	LLE	Local Structure	K-Nearest Neighbour Graph and Distance	No	Closed solution, Eigenvalue Problem
MVU	Weinberger and Saul (2006)	Isomap	Global Structure	K-Nearest Neighbour Graph and Distance	No	Closed solution, Eigenvalue Problem
SNE	Bunte et al. (2012)	-	Local Structure	Probabilities	No	Minimise Kullback-Leibler Divergence, local minima
t-SNE	Van der Maaten and Hinton (2008)	SNE	Local Structure	Probabilities	No	Minimise Kullback-Leibler Divergence, local minima
NerRV	Venna et al. (2010)	SNE	Local Structure	Probabilities	No	Minimise Kullback-Leibler Divergence, local minima

Table 2.1 Milestone Dimensionality Reduction Methods

## 2.2 Dataset Showcase

Before moving on to discuss those milestone DR methods, I am going to establish a dataset showcase because it is easier to understand what the algorithms really do with a concrete example and we also be able to compare their performance, using quality measures introduced in next chapter. Furthermore the showcase will help to raise meaningful questions about what is a realistic expectation from the scatterplot produced from DR methods and what is the gap between general dimensionality reduction and visualising high dimensional data as scatterplots.

The synthetic dataset is comprised of 1000 points, almost evenly distributed on a regular 3D spherical shell, shown by Fig. 2.1. This spherical shell is an interesting example to visualise because it is a closed and curved manifold embedded in 3D space. Pseudo-colours are given for each point according their z-axis value so that we can track the continuity of the sphere surface when the structure is visualised in 2D space.

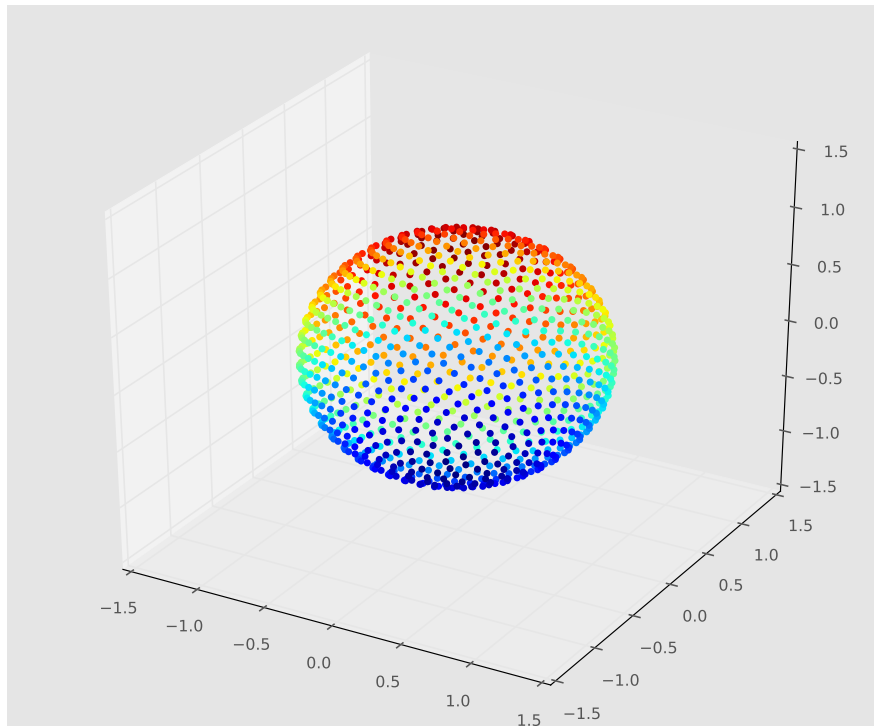


Fig. 2.1 The 3D Spherical Shell



## 2.3 Principal Component Analysis

### 2.3.1 Overview

Principal Component Analysis (PCA) is essentially a re-representation of the data via a linear transformation from its original basis into a new orthonormal basis (eigenvectors), in such a way that the variance among data on the projected axes is maximised, [Shlens \(2014\)](#). In particular, the first principal component (PC1) is the direction that captures the maximum variance of data; the second principal component does the same but has to be perpendicular to the previous one and so on. Dimensionality reduction is achieved by only selecting the first  $k$  principal components to represent the data. The goodness of using these eigenvectors to approximate original data is measured by the reconstruction error, or in other words, how much data variance is preserved by the new choice of basis. If the original data is  $m$  dimension and we let  $k = m$ , PCA is simply a coordinate change to a new orthonormal basis and hence the reconstruction error is zero. Though this idea of using “closest lines or planes to fit points in space” is initially proposed by [Pearson \(1901\)](#) over a century ago; it is perhaps the most important and widely accepted DR method today, because of its clear geometric intuition.

The input data of PCA,  $X$  is a  $n \times m$  matrix, in which each row is a point in  $m$ -space. Without loss of generality, we assume the column means of  $X$  are 0, so that the data is “centred” on the origin. PCA transforms  $X_{(n \cdot m)}$  into  $Y_{(n \cdot k)}$  via matrix  $U_{(m \cdot k)}$ , written as,

$$Y = XU$$

where  $U$  is an basis of  $k$  orthonormal vectors. According to the goal of PCA, the first column vector  $u_1$  should capture the maximum variance of  $X$ . Let  $\sigma_1$  be the sum of variance that  $X$  is projected onto  $u_1$ , written as,

$$\begin{aligned} \sigma_1 &= \frac{1}{n} \sum_{i=1}^n (u_1^T x_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n u_1^T x_i x_i^T u_1 \\ &= u_1^T \left( \frac{1}{n} \sum_{i=1}^n x_i x_i^T \right) u_1 \\ &= u_1^T S u_1 \end{aligned} \tag{2.1}$$

where  $S_{(m \cdot m)} = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$ , the covariance matrix of  $X$ . In order to maximise  $\sigma_1$ , one could solve it by forming the Lagrangian,

$$\sigma_1(u_1) = u_1^T S u_1 + \lambda(1 - u_1^T u_1)$$

and then by setting its gradient  $2S u_1 - 2\lambda u_1$  to 0, it reduces to following format,

$$S u_1 = \lambda u_1$$

Thus  $\sigma_1 = u_1^T S u_1 = u_1^T \lambda u_1$  and since  $u_1^T u_1 = 1$ , the maximum value of  $\sigma_1$  is  $\lambda$ . Here we found that  $u_1$  is the first eigenvector of  $S$  and  $\lambda$  is its corresponding eigenvalue. In linear algebra, PCA can be solved efficiently by finding eigenvectors of covariance matrix of  $X$  or in other words, the symmetric matrix  $S$  is diagonalised by a matrix of its orthonormal eigenvectors.

The reconstruction error is the sum of square error between original data vector  $x_i$  and estimated data vector  $\hat{x}_i$ , which is written as  $E = \frac{1}{n} \sum_{i=1}^n |\hat{x}_i - x_i|^2$ . The geometrical intuition of it could be explained as following. Suppose  $X$  is 2D data with its first and second feature highly correlated. As illustrated by Fig. 2.2, the first principal component (PC1) that PCA finds is the best-fit line to the data cloud. Since  $X$  is centred, the original variance of  $X$  is the sum of squared distance of each vector  $x_i$  to the centre point  $c$ , which is a constant value. Thus by maximising the projected variance of  $X$ , PCA automatically minimises the reconstruction error at the same time, as shown in Eq. 2.2. In fact the trace of the diagonalized covariance matrix is the sum of total data variance of  $X$ , and the sum of the projected variance is  $\sum_{j=1}^k \lambda_j$ . Thus the reconstruction error is also the sum of the least smallest eigenvalues  $\sum_{j=k+1}^m \lambda_j$ .

$$\begin{aligned}
 E &= \sum_{i=1}^n \sigma_i = \sum_{i=1}^n |x_i - c|^2 \\
 &= \sum_{i=1}^n (|x_i - \hat{x}_i|^2 + |\hat{x}_i - c|^2) \\
 &= \sum_{i=1}^n |x_i - \hat{x}_i|^2 + \sum_{i=1}^n |\hat{x}_i - c|^2 \\
 &= \text{Reconstruction Error} + \text{Projected Variance}
 \end{aligned} \tag{2.2}$$

### 2.3.2 Visualising Spherical Shell by PCA

PCA is usually employed as a standard dimensionality reduction procedure to pre-process or compress raw high dimensional data for further analysis. The scatterplot produced by PCA is a projection of the data onto the plain defined by the first two principal components. An unique “advantage” of PCA comparing with all other DR methods that I list above is that its projected

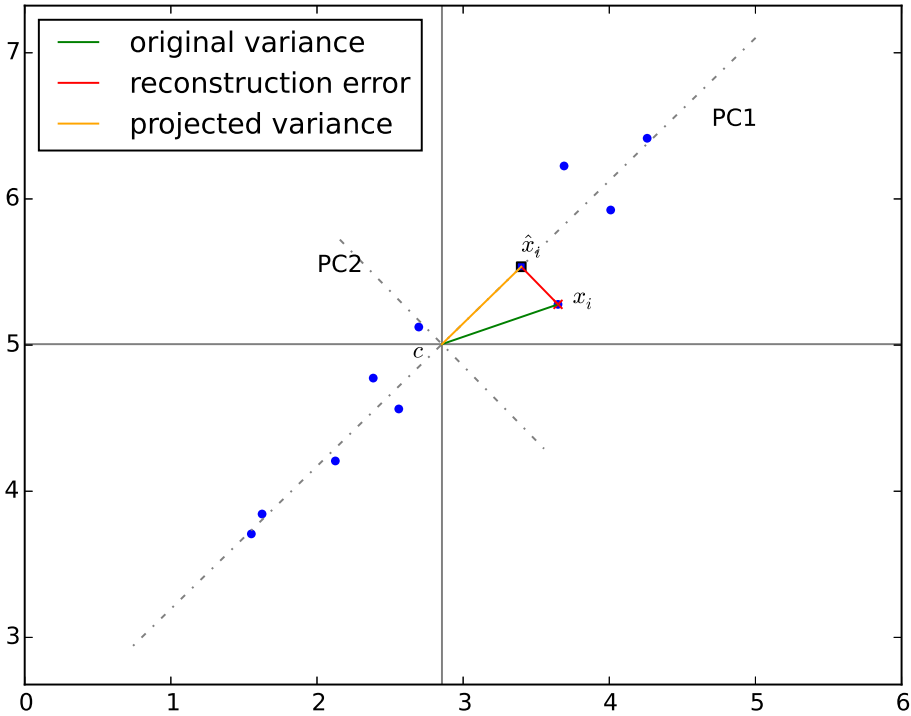
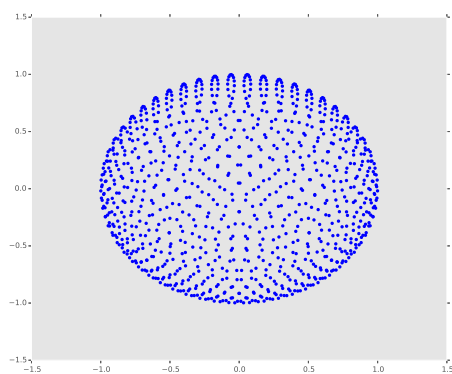


Fig. 2.2 Best Fit Line, The Intuition of PCA

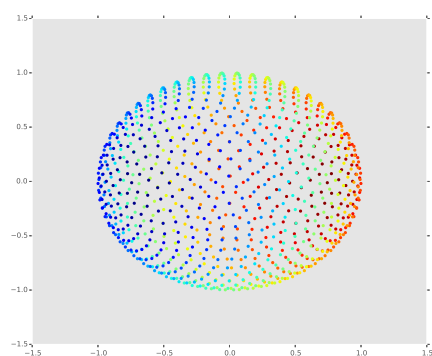
axes actually have geometric meaning. The main disadvantage of using PCA for scatterplot visualisation is that in many cases, the structure presented by its first two principal components will not be useful or faithful if the sum of its first two eigenvalues is only a small percentage of the trace of the diagonalized covariance matrix  $S$ ; in other words, the reconstruction error is high, when the intrinsic dimension of data  $X$  is more than 2 and no 2D plane can fit it well.

Since the points in Fig. 2.1 are evenly distributed over the surface of the 3D sphere, there is in principle no direction to project that has the largest variance. Hence applying PCA on this dataset would be no better than finding a random 2D plane to project onto. Nevertheless, Fig. 2.3a shows the 2D scatterplot produced by PCA applied on the sphere dataset. The first thing to confirm from Fig. 2.3a is that the dataset is projected as a circle, which reflects the fact it is linearly projected from a regular sphere. The point of showing the scatter plot without pseudo-colours here is that real world high dimensional data does not have any known geometric structure, and the plain scatterplot is only thing one could expect from the traditional high dimensional data visualisation.

Fig. 2.3b reveals the true story. In the middle part of the circle, the nearest neighbour of one point is not its nearest neighbour in original space. The simple explanation is that PCA gives a view of 3D sphere from the side directly, and it smashes the hollow sphere into a 2D dish. Thus points from one side of the sphere are mixed with the points of the other side.



(a) PCA scatterplot of 3D Sphere, no pseudo-colours.



(b) PCA scatterplot of 3D Sphere, in pseudo-colours.

Unless we have prior knowledge about the geometric structure of high dimensional data as in this simple case, PCA does not produce a 2D scatterplot suitable for inspecting local structure. What is a better way to visualise a dataset like the 3D sphere?

## 2.4 Beginnings of NLDR: MDS and Its Variants

Multidimensional Scaling (MDS) originated from an inverse problem – given a pairwise distance or dissimilar matrix and recover the coordinates of data points. This statement also suggests that MDS acts more like a graph embedding method. The approach MDS takes is to minimise a so called “stress” function, a sum of differences between the original pairwise distances  $d_{ij}$  and the corresponding reconstructed distances  $\hat{d}_{ij}$ . Today the term MDS refers to a family of DR algorithms that share the same feature of employing the stress function as their optimisation goal. MDS family generally is divided in three categories [Wickelmaier \(2003\)](#): Classical MDS; metric MDS; and non-metric MDS. Apart from the algorithm family of MDS and their variants, the idea of optimising difference between two configuration  $X$  and  $Y$  under certain consideration is very influential in modern state-of-art visualisation techniques as we shall see in later sections.

The following subsections will briefly describe Classical MDS, Traditional Metric MDS and Non-Metric MDS. However the more important variants Sammon Mapping and Curvilinear Component Analysis will be discussed in detail.

### 2.4.1 Classical MDS and Metric MDS

Classical MDS [Cox and Cox \(2000\)](#) (a.k.a. Classical Scaling or Principal Coordinate Analysis) is closely related to PCA. In PCA, the approximation coordinates  $Y$  are obtained from eigenvalue decomposition of the covariance matrix  $X^T X$ , and if we let  $Y$  have the same dimension of  $X$ ,  $Y$  is the full reconstruction of  $X$ . Given only the distance matrix (Euclidean metric is usually a natural choice)  $D$ , Classical MDS converts it into a scalar product matrix (or kernel matrix)  $XX^T$  using the trick of double centring and applies the similar method in order to recover  $X$  via  $Y$ . [Wickelmaier \(2003\)](#).

In linear algebra, there exists a matrix  $B$  such that  $A = B^T B$ , if  $A$  is a positive semi-definite matrix. The scalar product matrix  $K = XX^T$  is a semi-definite matrix. In order to obtain  $K$  only from Euclidean distance matrix  $D$ , Classical MDS apply “double centring” (subtracting both row mean and column mean) trick as follow,

$$K = -\frac{1}{2}JD^2J$$

where  $D^2$  is  $D$  element-wise squared,  $J = I - \frac{1}{n}ee^T$  and  $e$  is a column vector with all value of 1. The more detailed proof could be found in [Cox and Cox \(2000\)](#). Since  $K$  is the inner product matrix of  $X$ ,  $X$  can be recovered via singular value decomposition as follows:

$$K = U\Lambda U^T = XX^T$$

$$X = U\Lambda^{1/2}$$

where  $\Lambda$  is the diagonal matrix of eigenvalues of  $K$ . The associated embedding coordinates  $Y$  can be obtained using the top  $m$  eigenvalues and eigenvectors.

This result automatically minimises the “STRAIN” defined in Classical MDS:

$$STRAIN := \min_Y Tr(XX^T - YY^T)^2$$

and If only  $D$  is in Euclidean metric, it also automatically minimised the “STRESS” defined as,

$$STRESS := \sum_{i=1}^n \sum_{j=1}^n (d_{ij} - \hat{d}_{ij})^2$$

Note that in this sense, the result of Classical MDS is identical to PCA and in fact, it is the dual form of PCA, which is used when vector features are not available for PCA [Hastie et al. \(2009\)](#). However if the dissimilarity metric is not Euclidean, the results will be different from PCA. Thus Classical MDS and metric MDS have a broader use than PCA.

Non-metric MDS preserves distance order instead of distance itself, because its application area is when numeric similarity information is not directly available as in social science. It is slow, however non-metric MDS’s numeric optimisation approach inspired modern DR methods such as Sammon Mapping.

## 2.4.2 Sammon Mapping and Curvilinear Component Analysis

The main algorithms of the MDS family stand on their own. They are widely used in market research, psychology and social science, but perhaps their true legacy for modern DR and visualisation development is the numeric optimisation approach that measures the difference between  $X$  and  $Y$  under certain constraints and finding the approximation configuration  $Y$  that resembles  $X$  as much as possible [Kruskal \(1964\)](#); [Shepard \(1962\)](#).

[Sammon \(1969\)](#) proposed the non-linear mapping (NLM) for high dimensional data visualisation as a scatter plot, later known as Sammon Mapping. The idea is simple yet influential. It modifies the raw STRESS function of MDS, and uses gradient descent to minimise the error. The goal function defined as follows:

$$E = \frac{1}{\sum_{i < j} d_{ij}} \sum_{i < j}^N \frac{|d_{ij} - \hat{d}_{ij}|^2}{d_{ij}} \quad (2.3)$$

where  $d_{ij}$  is pairwise dissimilar metric (I will call it “distance” later on, unless causing confusion) of  $X$ ,  $\hat{d}_{ij}$  is for  $Y$ . Notice that the distance for inter-points are usually Euclidean but Sammon mapping allows different metric as well. One important feature is that by multiplying the weight  $1/d_{ij}$  to the distance difference, small HD distances would have more contribution to the cost function than large HD distances. Thus retaining the accurate match between small scale distance in visual space is given more emphasis than preserving large scale distance. As [Demartines and H  rault \(1997\)](#) pointed out, the weight function should be bounded; sometimes NLM’s emphasis on extremely small HD distance is too much, which leads to large or medium HD distances have very little effect on the optimisation.

Later [Demartines and H  rault \(1997\)](#) proposed Curvilinear Component Analysis (CCA). It was the time that neural networks and Self-Organising Map (SOM) were popular among the visualisation community, but in fact CCA is very similar to NLM in design. CCA also modifies the raw stress of MDS like NLM does in favour of local structure, but one important feature different from Sammon Mapping is that the weight function depends on visual space distance  $\hat{d}_{ij}$  rather than original space distance  $d_{ij}$ .

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} w(\hat{d}_{ij}, \lambda_y) |d_{ij} - \hat{d}_{ij}|^2 \quad (2.4)$$

where  $w(\cdot)$  is a weight function, it can be any bounded and monotonically decreasing function like Gaussian or Sigmoid. In the original manuscript it is a simple step function,

$$w(\hat{d}_{ij}, \lambda_y) = \begin{cases} 1 & \text{if } \hat{d}_{ij} \leq \lambda_y \\ 0 & \text{if } \hat{d}_{ij} > \lambda_y \end{cases}$$

where  $\lambda_y$  is the parameter to define local neighbourhood and at which the “unfolding” took place. Also CCA uses a specific version of stochastic gradient descent which makes it much faster than NLM, allowing quick feedback for user to adjust  $\lambda_y$  interactively.

Several studies – [Demartines and H  rault \(1997\)](#); [Lee et al. \(2002\)](#); [Lee and Verleysen \(2009\)](#) report that CCA outperformed NLM and metric MDS in many ways, though its cost function doesn’t have any global minimal guarantee. It is interesting because in terms of subjective visual clarity and objective visualisation quality metric local minimal methods tend to outperform global minimal methods in practice as I will discuss in later chapters.

### 2.4.3 Visualising Spherical Shell by NLM

NLM (Sammon Mapping) yields a similar result to PCA (Fig. 2.4), in which it maps two sides of the sphere into the 2D plane together instead of opening up the curved manifold. In the middle of the scatterplot, there is no smooth colour gradient, the nearest neighbour of every point in the visualisation is not its true nearest neighbour in the original space. NLM emphasises on preserving short pairwise distances by giving them more weight than the long ones, but still fails to preserve local structure in this case. It is interesting that even given an opened-up flat sphere as its initial layout, NLM still produces the same result. The only reason behind it is that the global minima of NLM's cost function doesn't regard the open-up flat sphere as the best in its own standard, thus it is not capable of opening up the 3D spherical shell. [Lee et al. \(2014\)](#) investigated the behaviour of DR methods from the mechanical system analogies and pointed out that NLM is not “plastic”, which means that the NLM cost function penalises “tearing up” (mismatch of small distances) so much that it can not open up the manifolds.

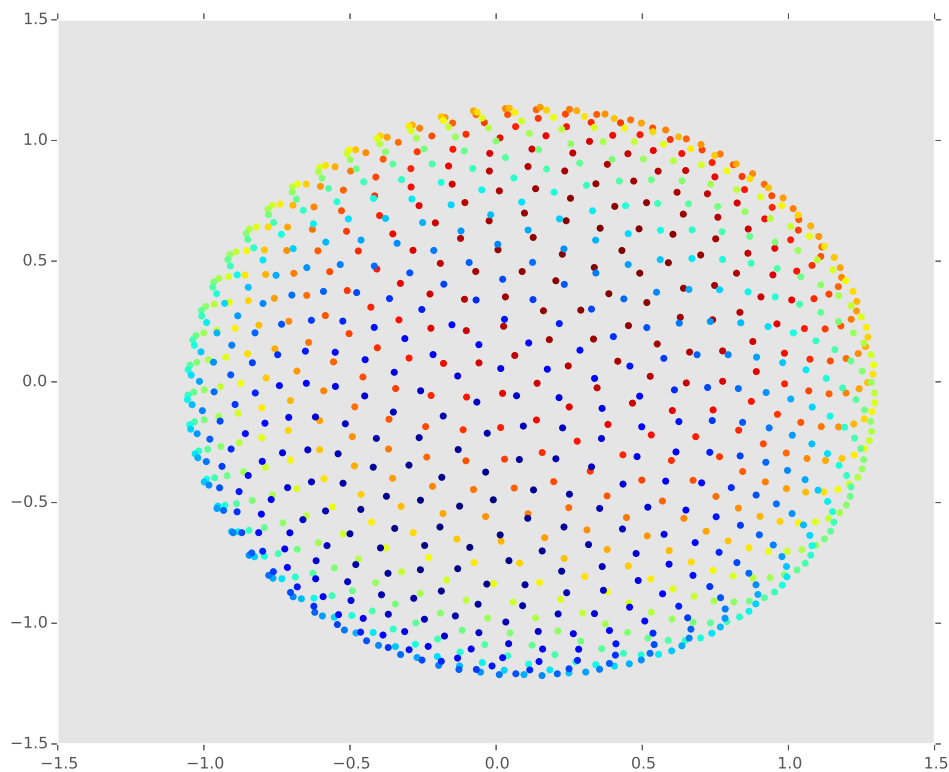


Fig. 2.4 NLM Visualisation of 3D Sphere in pseudo-colours



## 2.5 Isomap

The high dimensional visualisation research community has always been interested in visualising the folded manifolds embedded in HD space. One weakness of MDS and PCA is that the straight-line Euclidean distance will not reflect the true curved surface in the manifold. The better option is geodesic (curvilinear) distance, [van der Maaten et al. \(2009\)](#). Isomap [Tenenbaum et al. \(2000\)](#) tries to approximate the pairwise geodesic distance by computing the pairwise shortest path distance from a  $k$ -nearest-neighbour graph  $G$ , and then applies the approximated geodesic distance matrix  $D_G$  with MDS, since MDS has a guarantee of global minimality by performing eigenvalue decomposition. The procedure of Isomap is summarised as follows,

1. Build a Neighbourhood Graph: using  $k$ -nearest neighbours or a distance radius  $r$  to construct a fully connected neighbourhood graph  $G$ . Set every edge with weight  $d_{ij}$  based on the original distance metric.
2. Compute the shortest-path distance matrix: for every pair in  $G \{i, j\}$ , search its shortest-path and assign the shortest path distance  $d_{ij}^*$  to it, which will result with a new pairwise distance matrix  $D_G$ .
3. Apply Classical MDS on the geodesic distance matrix and obtain the low dimensional embedding configuration  $Y$ .

Since Isomap building a neighbourhood graph has become an essential procedure for modern DR methods. Though their sensitivities vary, all these algorithms suffer from inappropriate neighbourhood estimation. In the case of Isomap in particular, it is called “short circuit” or topological instability [Balasubramanian and Schwartz \(2002\)](#). If the neighbourhood is not carefully selected for even one pair, the estimated geodesic distance matrix might be completely different. In the original manuscript of Isomap (see [Tenenbaum et al. \(2000\)](#) footnote 18), it offers theoretical justification that if input data points are well sampled (with sufficiently high density), there is a bound for neighbourhood selection radius such that the estimated geodesic distance would be no longer than the true geodesic distance and small enough to prevent “short circuit”. However the input data is usually not well sampled in many cases, hence Isomap is noise-sensitive and the pre-conditions are hard to satisfy in practice.

### 2.5.1 Visualising Spherical Shell by Isomap

The  $k$ -nearest neighbour graph plays an important role in Isomap. Fig. 2.5a shows the visualisation of 3D spherical shell produced by Isomap with  $k = 2$ . It looks nothing like any other

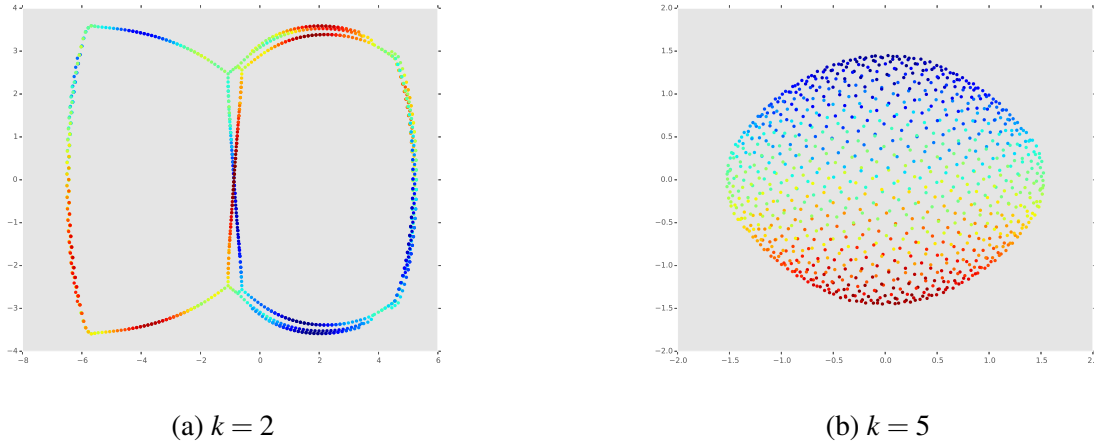


Fig. 2.5 Isomap Visualisation of 3D Sphere in Pseudo-colours

previous visualisation and its colour gradient is correct, but the parameter is inappropriate for the algorithm to show the structure. While setting  $k = 5$ , the geodesic distance built by 5-nearest neighbour graph nearly approximates the Euclidean distance, and it is not a surprise to see Isomap produces the similar result as PCA (Fig. 2.5b) since its third step is Classical MDS.

## 2.6 Locally Linear Embedding

Another manifold learning algorithm is Locally Linear Embedding (LLE). It only focuses on using linear models to reconstruct local structures and attempting to expand their inter-relationships into a global continuous configuration. In particular, a reasonable assumption LLE makes is that for a limited neighbourhood scope defined by  $k$  nearest neighbours or distance radius  $r$  in the HD space, these data points are distributed on the manifold that could be approximated by linear model [Saul and Roweis \(2003\)](#). Thus the first step of LLE is similar to that of Isomap: it builds a neighbourhood graph  $G$ . Next it uses a linear combination of all neighbours  $x_j$  of  $x_i$  to approximate it, by minimising the cost function:

$$E(W) = \sum_{i=1}^n |x_i - \sum_{j \in G(i)} w_{ij} x_j|^2, \quad \text{with } \sum_{j \in G(i)} w_{ij} = 1 \quad (2.5)$$

where  $W$  is the matrix collecting the reconstruction coefficients (weights)  $w_{ij}$  and  $G(i)$  is the set indexing all the nearest neighbours of  $x_i$  found by the first step. Note that the second step results with  $W$  encoded with local reconstruction information which is all the third step needs.

In the third and final step, LLE obtains the low dimensional coordinates  $Y$  by minimising following cost function, which is an eigenvalue problem. The two extra constraints are the centering and decorrelation which are similar to what PCA does:

$$\begin{aligned}
 E(Y) &= \sum_{i=1}^n |y_i - \sum_{j \in G(i)} w_{ij} y_j|^2 \\
 &= (Y - WY)^2 = Y^T (I - W)^T (I - W) Y \\
 &\text{with } \sum_{i=1}^n y_i = \mathbf{0} \text{ and } \frac{1}{n} \sum_{i=1}^n y_i y_i^T = I
 \end{aligned} \tag{2.6}$$

In the end,  $Y$  is found by computing the  $m$  (the number of dimensions of embedding space, 2 or 3 in visualisation subject) smallest eigenvalues of scalar product matrix  $(I - W)^T (I - W)$ .

### 2.6.1 Visualising Spherical Shell by LLE

LLE does not involve numeric optimisation like NLM and CCA, but performs a series of linear reconstructions for each point with its  $k$  nearest neighbours. Fig. 2.6a shows LLE traces the embedded 2D spherical surface through a line and constructs two overlapping “V” shapes with smooth colour gradient. While setting  $k \geq 5$ , LLE produces similar results to PCA (Fig. 2.6b). This is because the embedded 2D manifold is closed, the linear reconstruction of local neighbourhood will faithfully position local points regardless of the points from the other side of the sphere. Although this visualisation is globally optimal in respect of LLE, the visualisation does not outperform PCA in this case. It is also trivial to foresee that for  $k = 999$  (each point takes all other points into account for linear reconstruction), LLE will yield the exact same result as PCA.

## 2.7 Stochastic Neighbour Embedding and its variants

By 2002, some important studies in the high dimensional data visualisation community had established:

1. Wrapping HD distance with a weight function in NLM and CCA;
2. Numeric optimisation of cost function to find a low dimensional configuration  $Y$  to resemble  $X$  in non-metric MDS;
3. Shift of focus to preserving local structure instead of global in LLE.

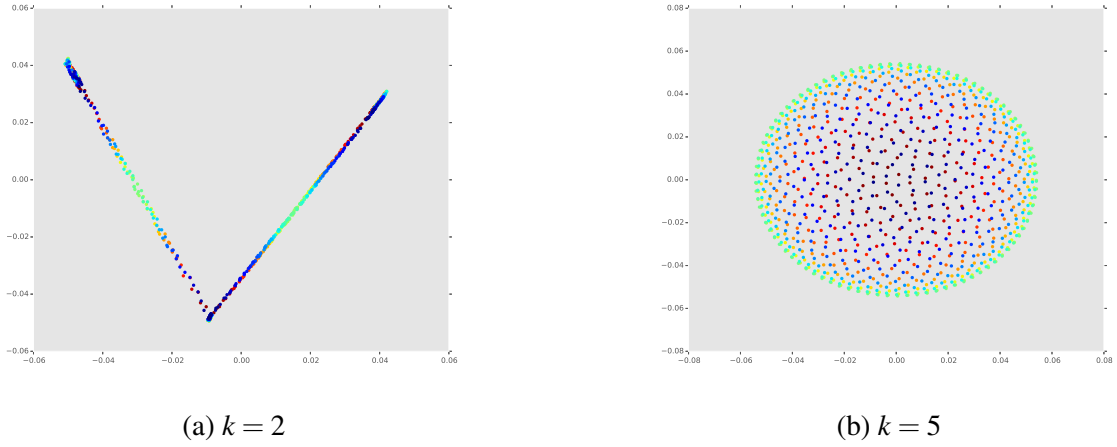


Fig. 2.6 LLE Visualisation of 3D Sphere

[Hinton and Roweis \(2002\)](#) took these inspirations and proposed an elegant approach called Stochastic Neighbour Embedding (SNE), which shone a new light in DR and HD visualisation community. Furthermore this algorithm was later revisited by [Van der Maaten and Hinton \(2008\)](#) and they developed a modified version called t-Distributed Stochastic Neighbour Embedding (t-SNE). SNE and t-SNE do share a lot of similarities, but t-SNE addresses some of the key practical issues of SNE. In my opinion, t-SNE is a truly ground-breaking work and has gained increasing popularity ever since.

The central idea of t-SNE is that it translates HD distances among data points ( $X$ ) into “affinities” via Gaussian Kernel and then constructs a global joint probability distribution ( $P$ ) to model the HD space. The low-dimensional configuration ( $Y$ ) is modelled with t-Distribution ( $Q$ ) in the similar fashion to  $P$ . The accumulated difference (the cost function  $C$ ) between two configuration  $X$  and  $Y$  is measured by KL-Divergence [MacKay \(2003\)](#) between the two distributions  $P$  and  $Q$ . By optimising  $C$  (see Eq. 2.11) with gradient descent, the appropriate  $Y$  is found.

The following set of equations are used in SNE and t-SNE, [Van der Maaten and Hinton \(2008\)](#). In particular, Eq. 2.7 is used as the conditional probability  $P$  for SNE, while Eq. 2.8 is the joint probability  $P$  for t-SNE; Eq. 2.9 is used as the low dimensional embedding distribution  $Q$  for SNE, and Eq. 2.10 is the corresponding  $Q$  for t-SNE. Finally Eq. 2.11 is the cost function used by SNE and t-SNE, though the only difference is that  $p_{ij}$  and  $q_{ij}$  should be replaced by  $p_{j|i}$  and  $q_{j|i}$  for SNE in respect of its conditional probability model.

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \quad (2.7)$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2} \quad (2.8)$$

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)} \quad (2.9)$$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \quad (2.10)$$

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \cdot \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (2.11)$$

Where  $\sigma_i$  acts as the density control for each  $p_{j|i}$ , in such ways that the entropy of every conditional probability  $\sum_{j \neq i} p_{j|i}$  is adjusted to be the same value. The detail of this will be discussed in Sec. 2.7.2.

### 2.7.1 Affinity instead of Distance

SNE translate the pairwise distance  $d_{ij}$  into conditional probability  $p_{j|i}$  in order to model the HD space  $X$ . There is no specific reason given in the original manuscript in neither SNE nor t-SNE why Gaussian kernel is used, despite [Venna et al. \(2010\)](#) adopts this definition which refers it as “a natural choice” for justification. Although the conversion does not make  $P$  a true probability distribution, it can behaviour like one so that the KL-Divergence can be applied. Here let us refer the numerator of  $p_{j|i}$  as “affinity” and denominator as the normalisation factor for the following discussion.

What is also interesting is that the bell-shell curve does provide a smooth shape for the affinity within 3-standard deviation region control by  $\sigma_i$ , namely close neighbour pairs have significant high affinity values than those are far apart. The embedding information  $\sum_{j \neq i} p_{j|i}$  for any point  $i$  is contributed by all other points  $j$ , especially its “close neighbours” with  $d_{ij} < 3\sigma_i$ . This is similar to LLE, in which the embedding information for each point  $i$  is a vector of linear coefficients of its close neighbours (see Eq. 2.5). Furthermore the way SNE records the embedding information is smoother than LLE, because every other point has an influence no matter how far away they are, even very insignificant.

One has to note that neighbourhood relationship is asymmetric. It is reasonable that SNE models how point  $j$  is close to point  $i$  as conditional probability, since the density of close neighbours within certain radius of point  $j$  could be different from point  $i$ , so that the two affinity values would not necessarily be the same. However this property is changed in t-SNE,

as it adopts the symmetric joint distribution instead of conditional. The effect of this will be explained in 2.7.3.

## 2.7.2 Perplexity as Soft Neighbourhood Selection

The only parameter of SNE or t-SNE is *Perplexity*. It could be considered as “the effective number of neighbours” for each data point [Van der Maaten and Hinton \(2008\)](#). Perplexity is directly responsible for the unsettled parameter  $\sigma_i$  in  $p_{j|i}$ . Formally perplexity is defined as follows:

$$\text{Perplexity} := 2^{H(P_i)} \quad (2.12)$$

where  $P_i = \sum_j p_{j|i}$  and  $H(P_i)$  is the Shannon entropy of  $P_i$  in bits, written as,

$$H(P_i) = - \sum_{j \neq i} p_{j|i} \log_2(p_{j|i}).$$

Since the equations can not be solved analytically, SNE performs a binary search for  $\sigma_i$  of each  $P_i$  such that every  $P_i$  is adjusted to have the same entropy of  $\log_2(\text{Perplexity})$ . In other words, it ensures  $\sigma_i$  is adjusted to be larger in sparse neighbourhoods and smaller in dense neighbourhoods.

There is no analytic explanation of the relationship between perplexity and  $\sigma_i$  in the original manuscript of SNE and t-SNE. To illustrate how *Perplexity* works, let us consider simple synthetic data distributed in 2D space with x-axis ranging from -4 to 4 and y-axis ranging from -3 to 3, as shown in Fig. 2.7. The data composites two wide separate clusters, but the left one (41 points of Gaussian data centred at (-2,0) with standard deviation of 0.5) is much denser than the right one (2 points of Gaussian data centred at (2,0) with standard deviation of 1.0). Let us name the left central red point as point  $a$  and the right central blue point as point  $b$ . In Fig. 2.7, the red dashed circles around the point  $a$  is the contour plot of the “affinity curve”  $\exp(-\|\hat{x} - x_a\|^2 / 2\sigma_a^2)$  and the blue dashed circles is the affinity curve for the point  $b$ . While the solid Gaussian curves at the bottom are just the projection of those two.

There are 24 points fall within the outermost red circle which is the border of  $3\sigma_a$  and 43 points for the point  $b$ , because  $\sigma_a$  (0.25) is smaller than  $\sigma_b$  (2.0). It illustrates that SNE adjusts the local  $\sigma$  for Gaussian kernel based on local density.

The projections of point  $a$  and point  $b$  that are marked on the solid Gaussian curve at the bottom of the diagram are the corresponding affinity value of the conditional probabilities  $p_{b|a}$  and  $p_{a|b}$  respectively. It shows that although the distance between point  $a$  and point  $b$  is fixed as

$d_{ab}$ , the level of how point  $b$  is close to point  $a$  is much different from the reverse case, which reinforces the fact that neighbourhood relationship is asymmetric.

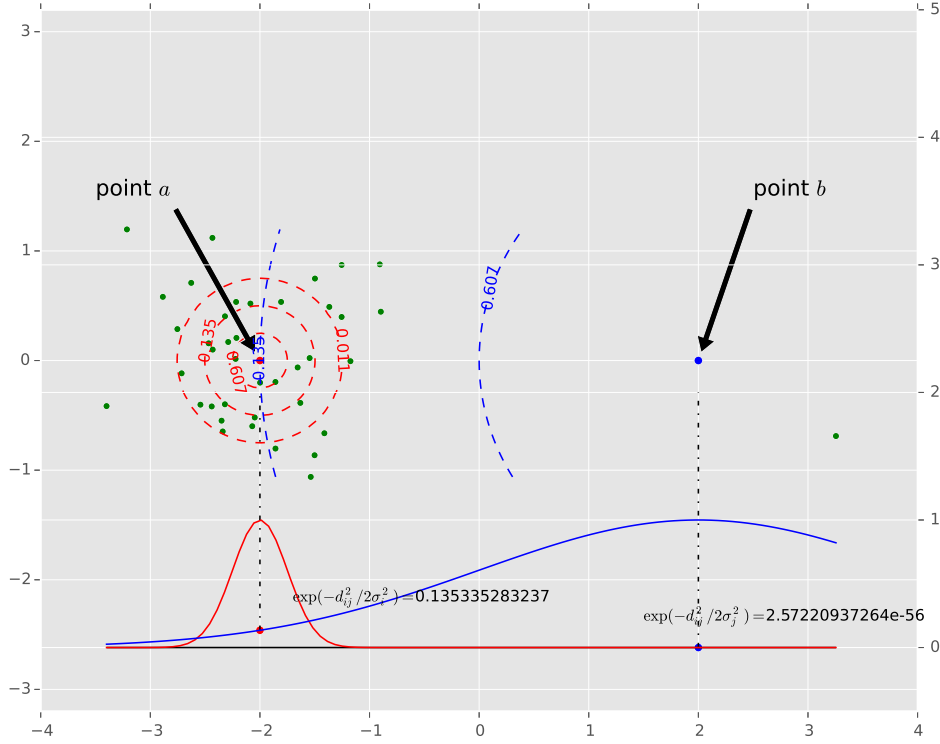


Fig. 2.7 Two Gaussian clusters in 2D, overlay with affinity curves (*perplexity* = 20).

### 2.7.3 Symmetric SNE

For a pair of points  $a$  and  $b$ , the values of the conditional probabilities  $p_{b|a}$  and  $p_{a|b}$  might be different in SNE, but t-SNE replaces it with joint distribution by taking the average of those two (see Eq. 2.8), so that the HD space distribution  $P$  becomes symmetric. The justification which Van der Maaten and Hinton (2008) states is that the gradient of the cost function of t-SNE (Eq. 2.14) is simpler than SNE's (Eq. 2.13), which is easier to optimise. And the paper also claims with a plausible tone that:

In preliminary experiments, we observed that symmetric SNE seems to produce maps that are just as good as asymmetric SNE, and sometimes even a little better.

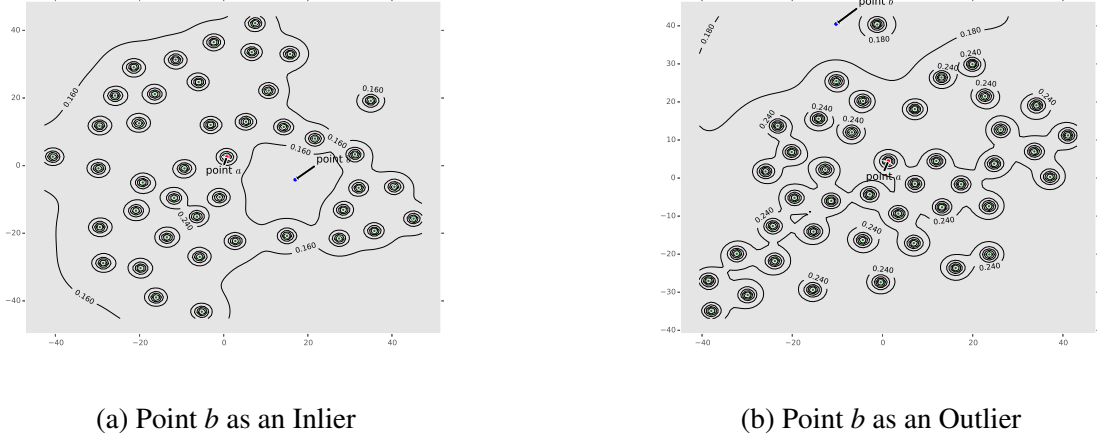


Fig. 2.8 Visualisation of data in Fig. 2.7 by t-SNE (*perplexity* = 20)

$$\frac{\partial C_{\text{SNE}}}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{i|j} + p_{i|j} - q_{j|i})(y_i - y_j) \quad (2.13)$$

$$\frac{\partial C_{\text{t-SNE}}}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} \quad (2.14)$$

For a detailed derivation, see appendix I in [Van der Maaten and Hinton \(2008\)](#).

But this benefit destroys the sense of the asymmetric neighbourhood relationships, in the way which the probability values of far-apart pairs become less contrast to the probabilities among the effective neighbours. Thus the position of an outlier is not well determined in the t-SNE visualisation, with some chances of being an “inlier” (an outlier that are pulled closely to some other points). Let us consider a visualisation experiment to demonstrate the effect of symmetric probability distribution.

Fig. 2.8a is a 2D re-projection by t-SNE to the 2D data shown in Fig. 2.7, with *perplexity* = 20. The cluster around point  $a$  is tight but is visualised sparsely and absorb the outlier point  $b$  as an “inlier”. The scatterplot is overlaid with the contour plot of the evaluation of the t-SNE cost function in the prospect to freely moving point  $b$  and fixing all other points’ position in the 2D visualisation. Point  $b$  is stuck in the “basin” right inside the cluster.

In Fig. 2.8b, another run of t-SNE under the same setting, t-SNE gives a reasonable position for point  $b$  as an outlier, although it is not as far as it used to be from point  $a$  originally. The contour plot in the diagram is the evaluation of the t-SNE cost function in the prospect to point  $b$ , showing that there is a “cliff” preventing point  $b$  from being close to the whole cluster.



However the vanilla SNE places point  $b$  as a distinctive outlier in Fig. 2.9 effectively. The three diagrams demonstrate the possible harm introduced by the symmetric distribution in t-SNE.

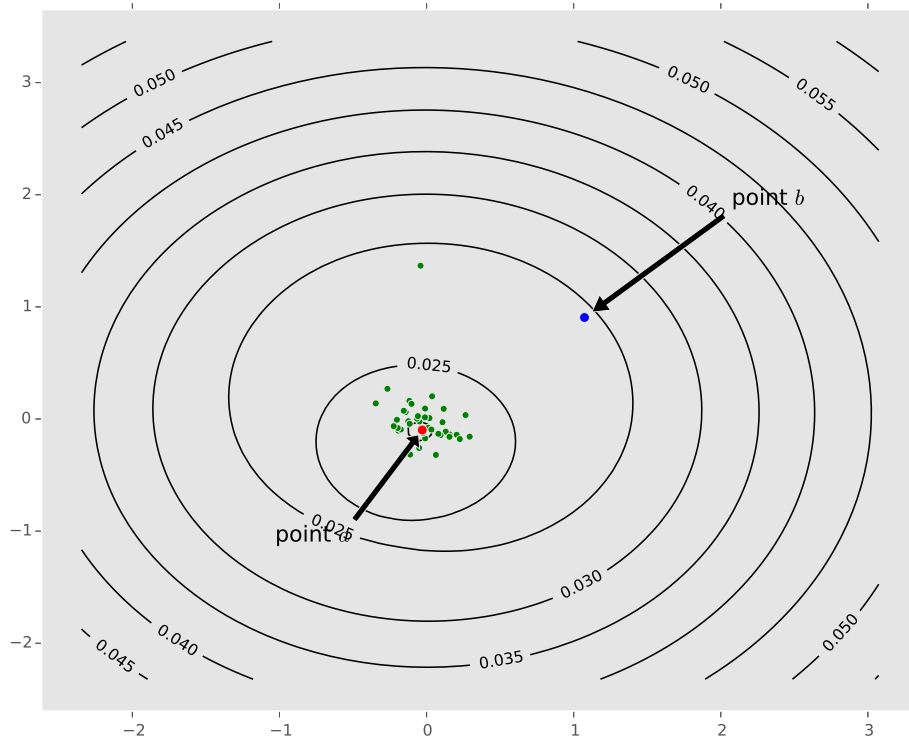


Fig. 2.9 Visualisation of data in Fig. 2.7 by SNE (*perplexity* = 40)

The inlier problem is harmful to scatterplot visualisation because there is no annotation for views to distinguish whether it belongs to a cluster or it is in fact an outlier. However this potential intrusion would not necessarily be a nemesis for visualisation if there exists an easy way to spot such errors. As we know, PCA would definitely visualise an outlier far away from other points and keep the original scale in data variance. But there is limited room in 2D space to place the outliers, because of this, the 2D space in PCA visualisation is “wasted” for presenting how far the outliers are while other structures are squeezed and unreadable. Later in the thesis I will introduce an overlay graph that corrects the inlier problem.

### 2.7.4 Heavy Tail Distribution

Another crucial difference between t-SNE and SNE is that instead of using a Gaussian to model the embedding coordinates (Eq. 2.9), t-SNE uses the t-Distribution with one degree of freedom (Eq. 2.10) [Van der Maaten and Hinton \(2008\)](#). The first feature of this choice is that pairs with moderate distances could be repulsed relatively further without affecting their probability mass too much, due to the heavier tail of t-Distribution. It is a compromise solution that alleviates a *practical issue* in SNE, which is known as the “crowding problem”.

The crowding problem is a general challenge that a complex manifold in HD space cannot be unfolded in 2D space by only utilising the pairwise distance information. Like Isomap using geodesic distance to approximate true manifold structure, t-Distributions make it possible that moderate distance pairs have some freedom to moving around in order to leave room for the local structure to unfold in 2D space. This feature also means t-SNE would choose to create some meaningfully tear-ups in favour of unfolding the manifold. Later in the thesis I will introduce an overlay graph to make it easier for viewer to discover such tear-ups.

### 2.7.5 Cost Function and Optimisation

The cost function of SNE and t-SNE is the KL-Divergence between distribution  $P$  and  $Q$  Eq. 2.11. The optimisation is achieved by gradient descent. However unlike previous DR methods, this cost function has one major disadvantage of being non-convex, thus it suffers from local minima and each run of the algorithm will produce a different scatterplot.

t-SNE treats the two different visualisation distortions differently. There is a relatively large positive cost if a pair of nearby points (high  $p_{ij}$ ) are put far apart (low  $q_{ij}$ ) in 2D space, which is a tear-up; But only a small or even possibly negative cost for representing a pair of far-apart data points (low  $p_{ij}$ ) with nearby layouts (high  $q_{ij}$ ), which is a false-neighbour. This asymmetric weighting scheme put preserving local structure as the priority. But this does not necessarily imply false-neighbours would happen more easily than tear-ups in t-SNE visualisation or that tear-up never happens. One may consider several scenarios:

**Points from tight cluster are hard to escape.** Suppose there is a tight cluster structure in the HD space. Any point find it hard to escape from the cluster in the visualisation, because there is a large positive cost for tearing up the point from its true neighbourhood. To demonstrate this argument, Fig. 2.10 is the 2D re-projection of the data in the previous example in Fig. 2.8a. The contour plot is the evaluation of the cost function of t-SNE in respect to freely moving point  $a$  (the centre of the cluster) and fixing all other points. The diagram shows that there is large positive cost for putting point  $a$  outside the cluster.

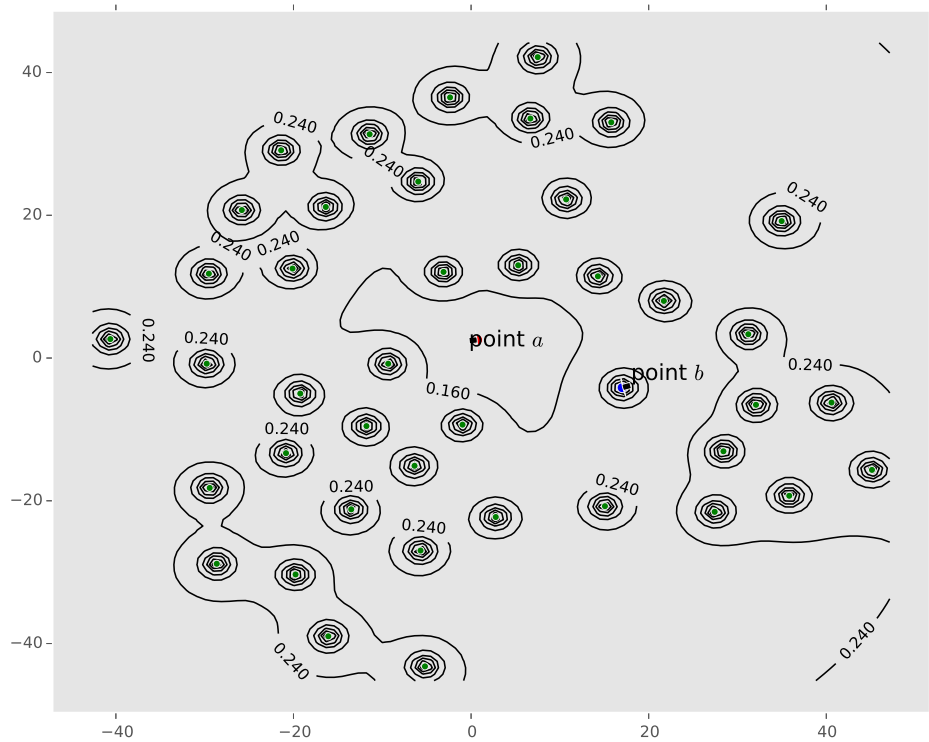
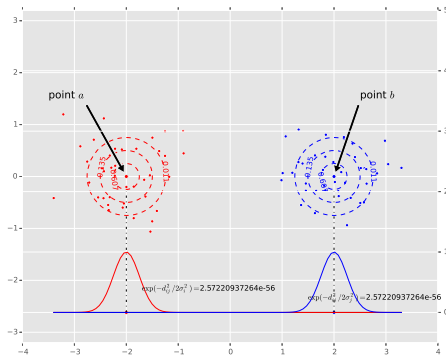
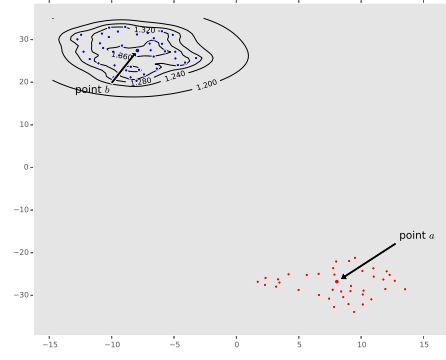


Fig. 2.10 The scatterplot in Fig. 2.8a; Contour Plot in Respect to Point  $a$ .



(a) Original 2D Layout.



(b) The 2D re-projection by t-SNE with Contour Plot of Moving Cluster A.

Fig. 2.11 The Twin Clusters Example.

**Positions of outliers are not well determined.** Because the t-SNE cost function does not punish representing far-away pairs by nearby positioning, a single point that is far away from any other points does not have a well determined position on the scatterplot. Thus during optimisation, an outlier can be pulled towards, or may even enter a strange neighbourhood in the 2D visualisation (Fig. 2.8a). Though the cost of putting it out of the strange neighbourhood in the first place may be even smaller (Fig. 2.8b), the local minimal is inevitable for the non-convex goal function.

**A cluster is hard to mix with another.** It is very unlikely for two far apart clusters to overlap, because if they overlap, the true neighbourhood within either cluster would result in more tear-ups than false-neighbours by contrast, hence the total cost is huge. To demonstrate this argument visually, let us consider an example of a twin clusters A (red) and B (blue) separated in the 2D space (shown by Fig. 2.11a). Fig. 2.11b is the 2D re-projection by t-SNE on this dataset, in which the twin clusters are well visualised. The contour plot in Fig. 2.11b is the evaluation of the t-SNE cost function in respect of moving cluster A as a whole towards cluster B. The contour lines suggest that cluster A can not move closer or overlap with cluster B, because the cost for doing so is huge.

**Some tear-ups are created to unfold structure.** Complex high-dimensional manifolds can not be naively flattened in the 2D space, but doing so will bring massive false-neighbours in the scatterplot. Because of the design of the cost function, the t-SNE optimisation would choose to create a few tear-ups and split the curved manifold in order to trade-off the larger cost sum by bringing massive false-neighbours.

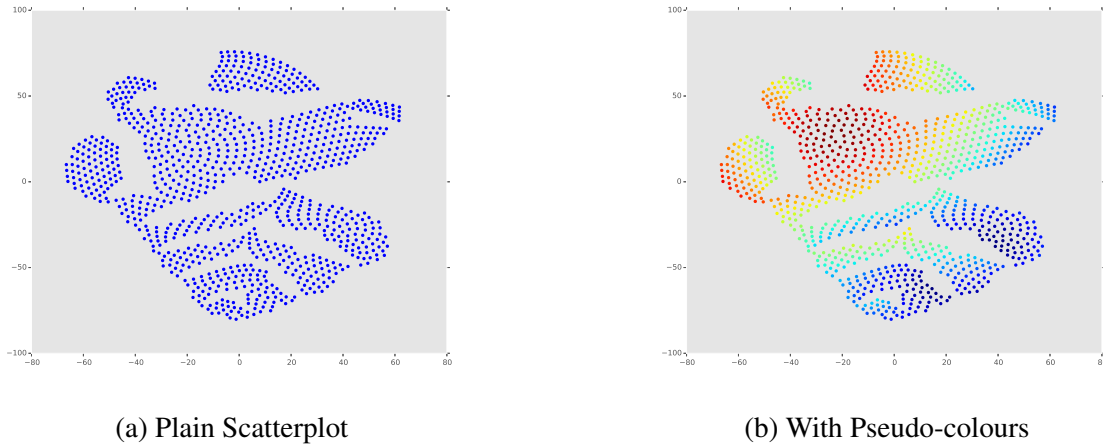


Fig. 2.12 t-SNE Visualisation of 3D Sphere (*perplexity* = 30)

### 2.7.6 Visualising Spherical Shell by t-SNE

The scatterplot (see Fig. 2.12a) produced by t-SNE with perplexity of 30 looks like no others. The visualisation has several patches but nothing indicates that it is projected from a 3D sphere. Fig. 2.12b has a smooth colour gradient across every patch, which perhaps implies the patch is one piece of the 3D sphere surface. But this might not be true, because psychologically our eyes automatically seeking for the simplest explanation for a visual pattern.

Lastly, I briefly present the effect *perplexity* on t-SNE visualisation. Here we choose a series of perplexity, e.g. 5, 10, 60, 120, 240, 900 for testing, shown in Fig. 2.13. The bigger the perplexity is the bigger the patches are. An interesting case is, when the perplexity is nearly equal to the number of data instance, e.g. 900, the scatterplot is almost identical to what PCA does (Fig. 2.13f).

## 2.8 Conclusion

This chapter reviews several milestone high dimensional data visualisation techniques from dimensionality reduction. Each of them has its own successful applications on visualising some toy dataset or real world HD data. The traditional visualisation of the HD data is nothing more than a plain scatterplot, in which many distortions can happen. So it is crucial to know how well the visualisation is, which leads to the next research area – the objective quality metric.

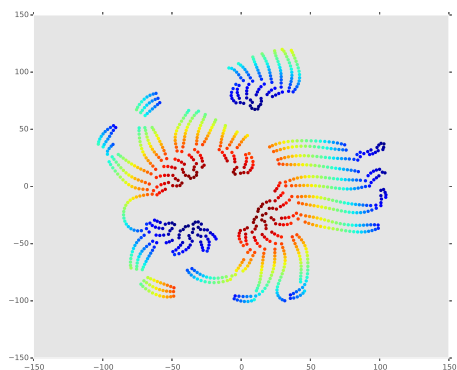
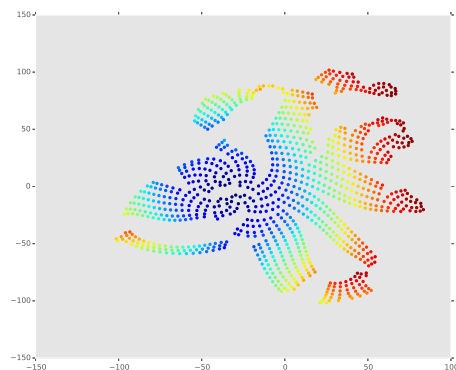
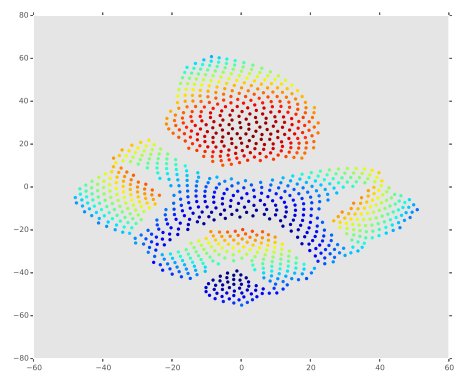
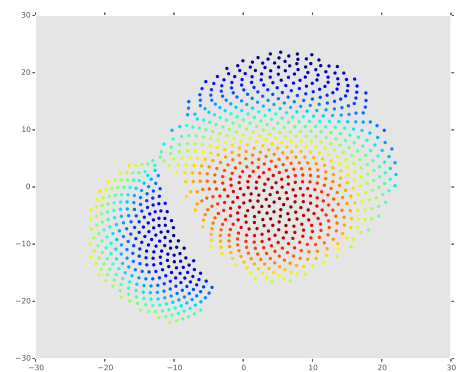
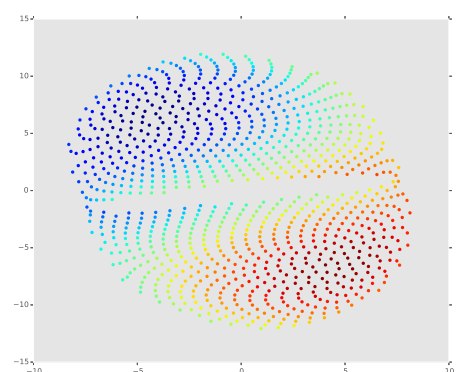
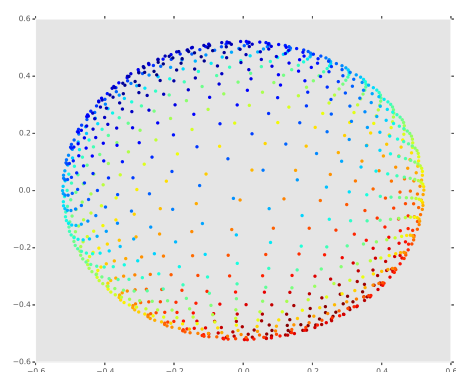
(a) *perplexity* = 5(b) *perplexity* = 10(c) *perplexity* = 60(d) *perplexity* = 120(e) *perplexity* = 240(f) *perplexity* = 900

Fig. 2.13 t-SNE Visualisation of the 3D Sphere in Different Perplexity.

# Chapter 3

## Scatterplot Visualisation Quality Assessment

### 3.1 Overview

Tear-ups and false-neighbours errors are unavoidable in the visualisation produced by any dimensionality reduction method, thus the next natural research interest is to assess the quality of the scatterplot objectively. Generally each DR method is designed as an optimisation problem. It is possible to measure the visualisation quality *globally* by looking at their optimisation performance. The list below shows possible algorithm-specific quality measures:

- Linear Method like PCA can be assessed by its *reconstruction error*;
- MDS family methods like NLM, CCA and Isomap can be assessed by their *stress-function*;
- SNE family methods like SNE and t-SNE can be assessed by their *cost-function* error.

In particular, for the last two, the accumulated optimisation error can be decomposed and allocated to each data point. The error for each individual data point is an indicator of “comfort” of how well it is fitted in the scatterplot *locally*. It is an obvious extension but not widely seen in the literature. However these global and local quality measures are only algorithm specific. It is difficult to use them as a touchstone for the visualisation produced by any DR method.

The centre idea of the scatterplot visualisation of HD data is that we expect the proximities in the plot reflect the corresponding similarities of the pairs in the HD space, within which the two arch distortions tear-up and false-neighbour are recognised. In this chapter, I will review some of the most important traditional quality measures that regards the scatterplot of HD data as a general task and focuses on the two kinds of error involved.

## 3.2 Shepard Plot and Co-Ranking Histogram

Shepard Plot was introduced in Non-metric MDS by [Shepard \(1962\)](#). It is a scatterplot or 2D histogram of the correlation between the pairwise input distance  $d_{ij}$  (HD space) and the pairwise output distance  $\hat{d}_{ij}$  (2D space). It is a model-free presentation of the performance of MDS family algorithms since their goal is about minimising the difference between  $d_{ij}$  and  $\hat{d}_{ij}$ . It also means the output distance has to be in the same scale of the input distance, which is a feature of MDS family algorithms all share. Generally in the Shepard plot, the x-axis is for the input distance and the y-axis is for the output distance. There are three parts in the Shepard plot to consider:

- The dots on the diagonal line are the pairs of points with their output distances match their input distances exactly;
- The dots in the lower triangle correspond to the pairs that are visualised closer in than they are in the HD space (false-neighbours);
- The dots in the upper triangle are the pairs that are placed further apart in the visualisation than they are in the HD space (tear-ups).

The Shepard plot is an useful detailed inspector for MDS family algorithms. The further the dots are away from the diagonal line the worse the pairs are distorted. It can not easily applied on SNE and t-SNE because the scale of output distance are independent and inconsistent from the input distance.

However one has to note that HD distance and 2D distance are only one interpretation of the similarities and proximities in the visualisation task. The Co-Ranking Histogram of [Lee et al. \(2008\)](#) is similar to the Shepard Plot but uses distance ranks instead of actual distances, namely the x-axis represents for HD distance ranks  $r_{ij}$  and the y-axis is for 2D distance ranks  $\hat{r}_{ij}$ . Similarly, there are three parts in the Co-Ranking Histogram to consider:

- The dots on the diagonal line are the pairs of points with their output distance ranks equal to their input distance ranks;
- The dots in the lower triangle are the pairs with their output distance ranks smaller than their input distance rank (false-neighbours);
- The dots in the upper triangle are the pairs of points with their output distance ranks larger than their input distance ranks (tear-ups).



The Co-Ranking histogram overcomes the possible scale difference between the 2D and HD distances, but loses the sense of local densities. The further the dots stay away from the diagonal line the worse they suffer from the distortion.

Fig. 3.1b and Fig. 3.2b are the collection of the Shepard plot and Co-Ranking Histogram corresponding to the scatterplot of the 3D Spherical Shell produced from DR methods reviewed in Chapter 2. Previously we know that PCA, NLM, Isomap and LLE give the similar visualisation because the two sides of the 3D sphere are overlapped and flattened on the 2D space. Thus their Shepard plots and Co-Ranking histograms are almost identical. In particular, their Shepard plots suggest that there are great number of pairs that are visualised closer than their original distances, which means massive false-neighbours distortion. And because of this, the close neighbours for each points are squeezed out (tear-ups in terms of distance ranks), as the arch above the diagonal line in their Co-Ranking histograms indicate.

However even with Shepard plot and Co-Ranking histogram we still do not know how the distortions happen and where they are in the scatterplot. In Chapter 2, we see t-SNE gives a different visualisation to the 3D Sphere, but its Shepard plot and Co-Ranking histogram suggest it suffers from both tear-ups and false-neighbours. We need other quality assessment to see whether it is true or not.

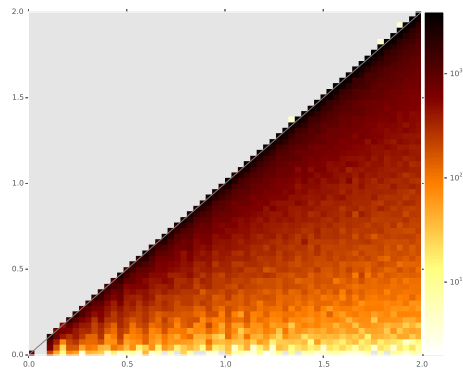
### 3.3 Venna's Trustworthy and Continuity

Trustworthy and Continuity (Venna's T & C) are a pair of distance-rank based quality measures for the scatterplot visualisation of the HD data. In particular, *Trustworthy* describes the level of false neighbours enter the finite neighbourhood of one point in the output space. While *Continuity* summaries the level of true neighbours leaving the neighbourhood of one point in the output space. The size of the neighbourhoods of input and output space are defined by an "arbitrary small" number of  $k$ . Both criteria are normalised between 0 and 1, the higher they are the better the visualisation quality is in certain respect. The specific definitions are given as follows Venna and Kaski (2001):

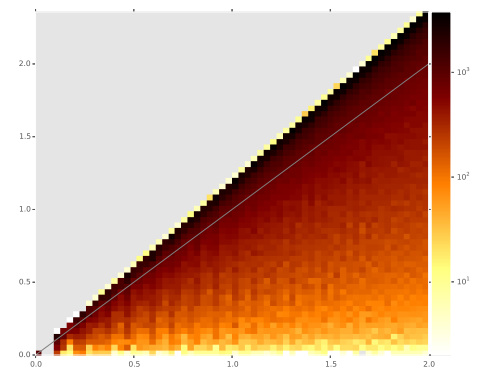
$$T(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^N \sum_{x_j \in U_k(x_i)} (r(x_i, x_j) - k) \quad (3.1)$$

$$C(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^N \sum_{x_j \in V_k(x_i)} (\hat{r}(x_i, x_j) - k) \quad (3.2)$$

- $x_i \in \mathbb{R}^n, i_1, \dots, N$  data vector in input space;



(a) PCA



(b) NLM

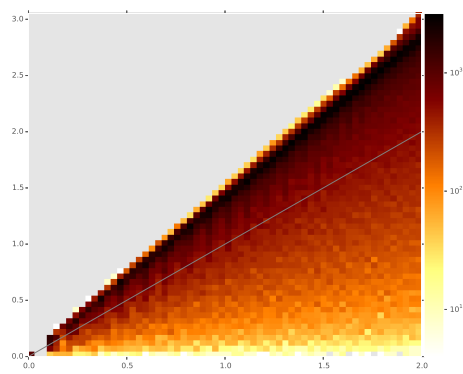
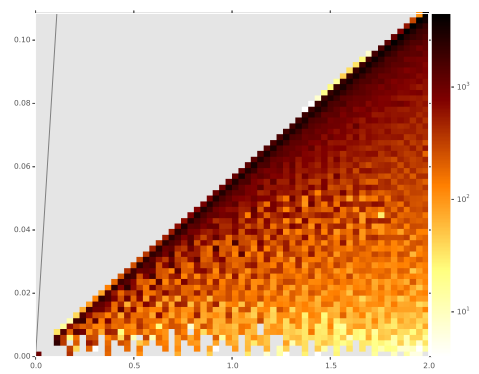
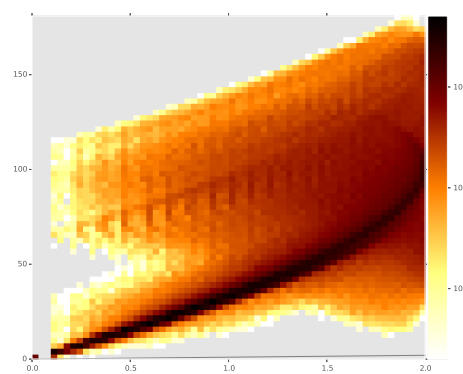
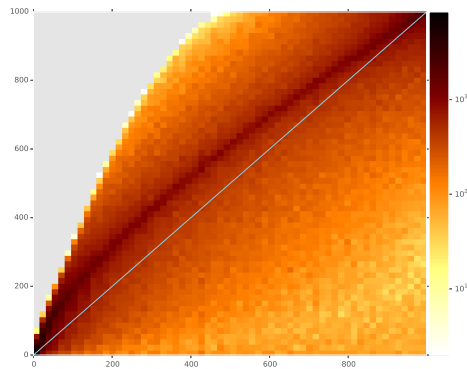
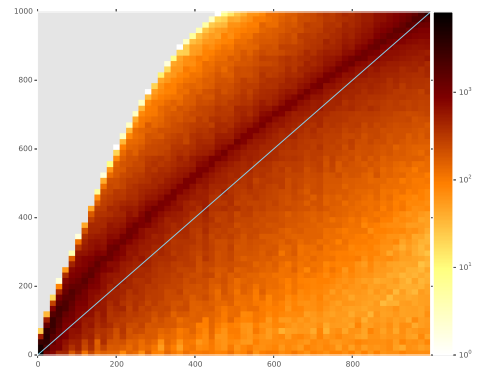
(c) Isomap ( $k=5$ )(d) LLE ( $k=5$ )(e) t-SNE ( $\text{perplexity}=30$ )

Fig. 3.1 Shepard Plot of the DR visualisation of the 3D Sphere



(a) PCA



(b) NLM

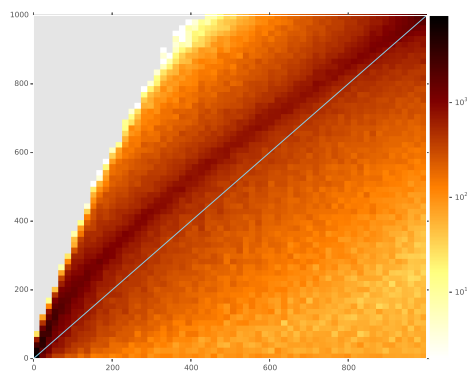
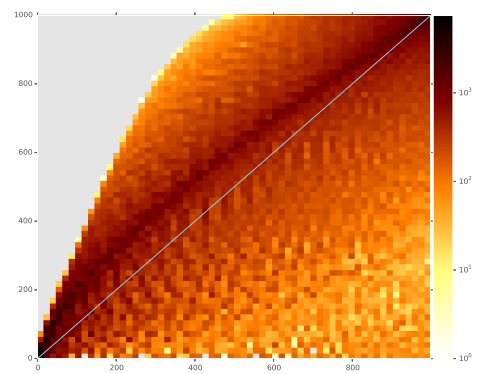
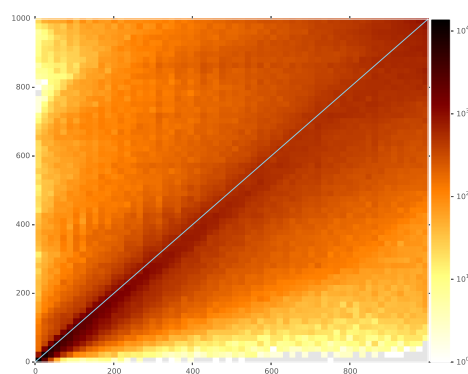
(c) Isomap ( $k=5$ )(d) LLE ( $k=5$ )(e) t-SNE ( $\text{perplexity}=30$ )

Fig. 3.2 Co-Ranking Histogram of the DR visualisation of the 3D Sphere

- $C_k(x_i)$  the set of those  $k$  data vectors that are nearest to  $x_i$  in the input space;
- $\hat{C}_k(x_i)$  the set of those  $k$  data vectors that are nearest to  $x_i$  in the output space;
- $U_k(x_i)$  the set of data vector  $x_j$ , such that  $x_j \in \hat{C}_k(x_i) \wedge x_j \notin C_k(x_i)$ ;
- $V_k(x_i)$  the set of data vector  $x_j$ , such that  $x_j \notin \hat{C}_k(x_i) \wedge x_j \in C_k(x_i)$ ;
- $r(x_i, x_j), i \neq j$  the neighbour rank of  $x_j$  to  $x_i$ , when the data vector are ordered based on their pairwise Euclidean distance in the input space;
- $\hat{r}(x_i, x_j), i \neq j$  the neighbour rank of  $x_j$  to  $x_i$ , when the data vector are ordered based on their pairwise Euclidean distance in the output space.

Trustworthy and Continuity (T& C) was introduced in 2001, the time when MDS family methods were popular as visualisation techniques. The original paper [Venna and Kaski \(2001\)](#) of T & C is among the earliest studies of relating the scatterplot visualisation of HD data to human observation, which is a visionary research direction.

Trustworthy is essentially the measure of false-neighbours and Continuity is for tear-ups. The paper remarkably points out that false-neighbour errors are “even more harmful” than tear-up errors, because false-neighbours reduce the trustworthy of the local proximity in the visualisation. I agree with this point and prefer those DR methods that produce fewer false-neighbour errors than the other one.

### 3.3.1 Using Venna’s T & C to Measure Visualisation of The 3D Sphere

Fig. 3.3 is the Trustworthy & Continuity performance plot of the visualisation produced by 5 DR methods (parameters are same as in Fig. 3.1) to the 3D sphere dataset mentioned in Chapter 2. The size of HD and 2D neighbourhood value  $k$  is given as a series of numbers ranging from 10 to 200. As discussed in chapter 2, PCA, NLM, Isomap and LLE give the similar scatterplots on this dataset, which explains their overlapping T & C curves. The curves are also consistent with the their Shepard plots and Co-Ranking histograms, in which massive false-neighbours are identified.

In contrast, the trustworthy of t-SNE visualisation remains the highest, nearly **1.0** with  $k = 10$ , which means for a small neighbourhood, there is rarely no other false-neighbours. However t-SNE also remains the lowest in continuity measure. This means there are more tear-ups in the visualisation than any other methods.

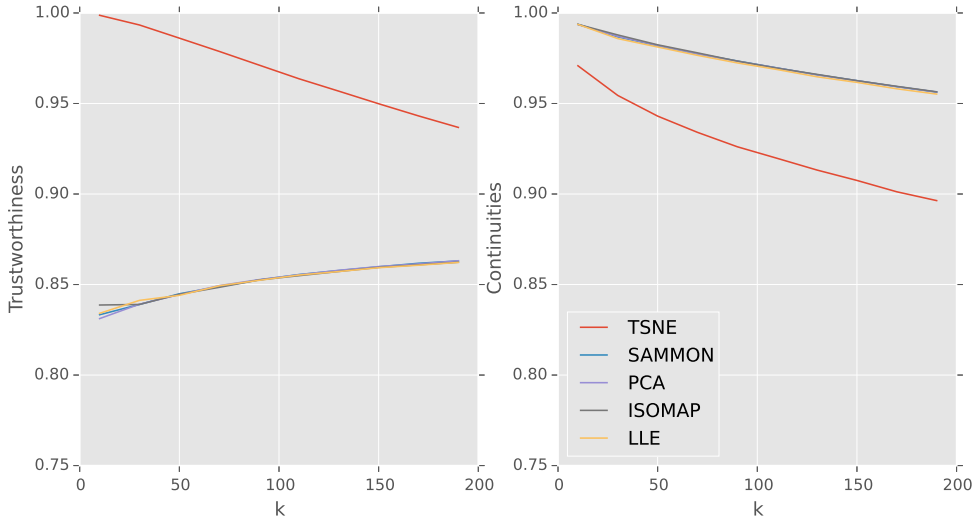


Fig. 3.3 Trustworthy and Continuity Performance for DR Methods to The 3D Sphere

### 3.4 Venna's Precision and Recall

Venna et al. (2010) gave a fresh look at the HD data visualisation in Information Retrieval prospective. The main argument is that no matter which DR method is used to produce the scatterplot, the performance assessment should be as a task of information retrieval problem, in which the relationship between mismatch of pairwise distances are examined in the context of finite local neighbourhood.

In a binary classification problem of Information Retrieval Theory, *Precision* is the fraction of retrieved items that are relevant; While *Recall* is the fraction of relevant items that are retrieved. In Venna's Precision and Recall for the HD data visualisation, the relevant items are those considered as being "close" to an individual item  $i$  in the HD space (input neighbourhood of data point  $i$ , noted as  $C_i$ ); While the retrieved ones are those visualised close to a point  $i$  in the 2D space (output neighbourhood of data point  $i$ , noted as  $\hat{C}_i$ ). The local neighbourhoods can be defined in variety of ways, such as a fixed number of nearest neighbours or a distance radius. The thresholds can be global or point specific. Generally let  $r_i$  be the number of items in  $C_i$  and  $k_i$  be the number of items in  $\hat{C}_i$ . The true-positive (hit) items are those relevant item in the retrieved set, denoted by  $N_{TP,i}$ . Then precision and recall for a single *query* for a data point  $i$  in the visualisation by "naked eye" are defined as:

$$\text{precision}_i = \frac{N_{TP,i}}{k_i} = 1 - \frac{N_{FP,i}}{k_i} \quad (3.3)$$

$$\text{recall}_i = \frac{N_{TP,i}}{r_i} = 1 - \frac{N_{Miss,i}}{r_i} \quad (3.4)$$

where  $N_{FP,i}$  is the number of false-positive items and  $N_{Miss,i}$  is the number of missed item.

Eq. 3.3 is the basic definition of Venna's Precision and Recall (P & R). It is similar to Venna's T & C but the neighbourhoods in HD and 2D are allowed to be different and they are not necessarily defined by distance ranks alone. Thus Venna's P & R has a more general term. A typical usage of Venna's P & R to measure the scatterplot visualisation quality is listed as following steps:

1. The input neighbourhood  $C_i$  for each point  $i$  must be searched using any preferred method at first. Then the size of each neighbourhood  $r_i$  is obtained.
2. The size of output neighbourhood  $k_i$  can be within a reasonable testing range like from 1 to 50 nearest neighbours in the 2D space.
3. When the size of input and output neighbourhoods are defined, a pair of  $\text{precision}_i, \text{recall}_i$  is obtained for each data point  $i$ .
4. The global visualisation quality *mean-precision* and *mean-recall* measure are the average of all point specific precision and recall values.
5. A sequence of mean precision and recall are generated given a sequence of testing values of  $k_i$ . Typically mean-recalls are drawn against mean-precisions to form a curve. The higher the start point of the curve is, the better precision of the retrieval; The later the curve drops, the better the capability of the retrieval.

The point specific precision tells an user the level of how many true neighbours of one data point are still within its output neighbourhood; While the point specific recall tells the level of how many points in the output neighbourhood are actually within input neighbourhood. Thus the quality of the scatterplot is viewed as measuring the capability of the output neighbourhoods ( $\hat{C}_i$ ) for retrieving the "items" in the input neighbourhoods. The two kinds of distortion Tear-up and False-neighbour are defined in Venna's P & R with different names. The *False-positives* are false-neighbours and the *Miss* items are tear-ups.

It is important to note that Venna's P & R is not useful if the size of the output neighbourhoods are too large. Firstly a large  $k_i$  value will definitely yield a high recall value, but it is not

an indication of good visualisation capability. Secondly a large  $k_i$  value is counter-intuition, because our naked eyes may not be able to observe a large number of points as neighbourhood in the scatterplot and it also gives a meaningless low precision performance.

The sizes of the input neighbourhoods are related to the application, but the sizes of the output neighbourhoods are directly linked to our capabilities of reading the diagram. If they are not appropriately defined, the performance figures are misleading. Usually one can use the parameter of the DR methods as a guideline for the size of the input neighbourhood. For example if letting 10 to be the  $k$ -nearest neighbours for LLE/Isomap or the perplexity for SNE/t-SNE, the input neighbourhood size can be set as 10 in metric of distance rank globally. The experiments in [Venna et al. \(2010\)](#), the size of the output neighbourhood is set to be in the range between 1 and 100 in the metric of 2D distance rank. However the large output neighbourhood size is not related to human perception, this “default setting” deviates from the paper’s original intention in my opinion.

### 3.4.1 Using Venna’s P & R to Measure Visualisation of Spherical Shell

Fig. 3.4 is the Precision and Recall performance plot of the previous DR methods on visualising the 3D sphere mentioned in Chapter 2. The global input neighbourhood size  $r$  is 30 (consistent with the default parameter of t-SNE) and the global output neighbourhood size  $k$  is a series testing value ranging from 1 to 60.

The top diagram in Fig. 3.4 is the mean-precision v.s. mean-recall plot, in which the higher and longer the curve stays on the top the better the quality the visualisation has. The curves suggest that t-SNE has an exceptional performance over other 4 DR methods the prospective of information retrieval. The bottom left diagram is the precision performance against the change of the size of the output neighbourhood. The curves suggest that t-SNE is superiors than the other 4 methods in the precision measure, especially when  $k$  is small. The bottom right diagram is the recall performance against the change of  $k$  value. Note that this diagram is only valid when  $k \geq 30$ , because the retrieval items are only a fraction of relevant items if  $k$  is smaller than  $r = 30$ . The diagram tells that recall value of t-SNE remains the highest and nearly a constant when  $k \geq 30$ , which means the method is not sensitive to the change of testing size of the output neighbourhood.

## 3.5 CheckViz

The idea of using a decomposed global visualisation quality measure as a point-specific quality metric is not new (see Section. 3.1). Colouring the scatterplot based the values each point

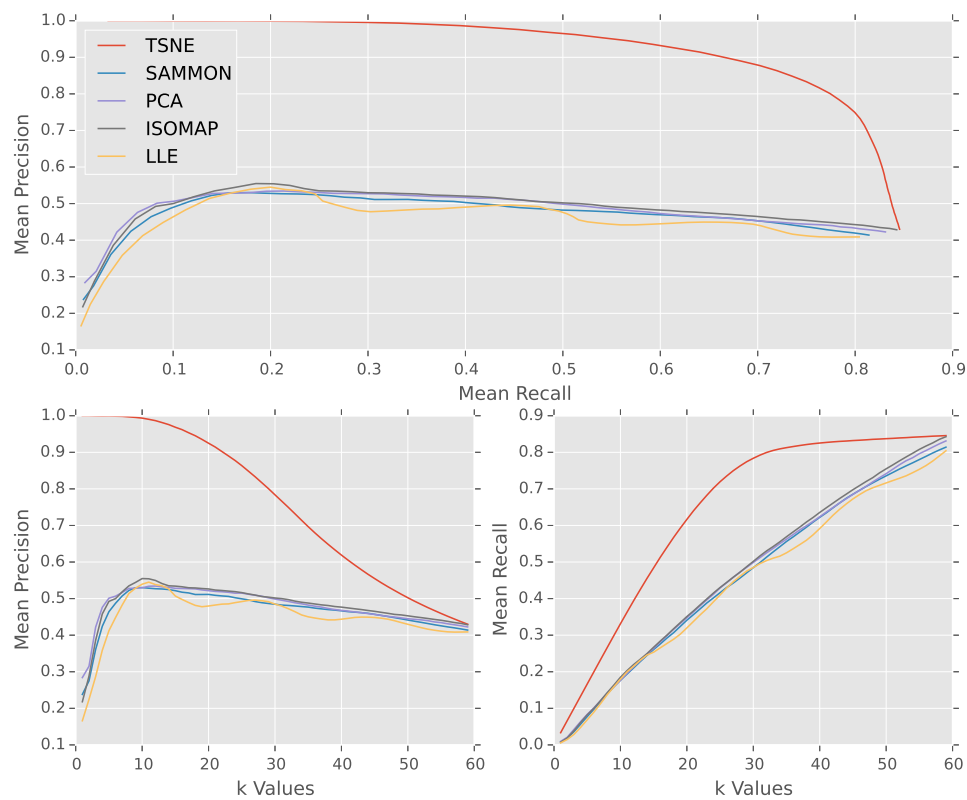


Fig. 3.4 Mean-Precision against Recall for DR Methods to the 3D Sphere. (input neighbourhood size  $r = 30$ )



contributes to the global error could be potentially beneficial because one would be able to gain some knowledges of which parts suffer the distortion the most. CheckViz [Lespinats and Aupetit \(2011\)](#) uses a pair of point-specific measures,  $(S_{CCA}(i)$  and  $S_{NLM}(i))$  to indicate the local distortion directly on the visualisation. The two local measures are encoded on the *Voronoi Cell* corresponding to each point in the scatterplot with the help of a *two-dimensional perceptually uniform colour table*. Any point in the scatterplot could suffer from both tear-up or false-neighbour distortions. With the level of distortion defined by  $(S_{CCA}(i)$  and  $S_{NLM}(i))$ , the pair of measures gives an individual point  $i$  a 2D coordinate on the 2D colour map.

The definitions of the two measures are as follows:

$$S_{NLM}(i) = \sum_j (\|d_{ij} - \hat{d}_{ij}\|^2 \times F_{\sigma}(d_{ij})) \quad (3.5)$$

$$S_{CCA}(i) = \sum_j (\|d_{ij} - \hat{d}_{ij}\|^2 \times F_{\sigma}(\hat{d}_{ij})) \quad (3.6)$$

Where  $F(x)_{\sigma}$  is a step function and  $\sigma$  here serves as a global distance radius threshold:

$$F(x)_{\sigma} = \begin{cases} 1 & \text{if } x < \sigma \\ 0 & \text{otherwise} \end{cases}$$

In Eq. 3.5,  $S_{CCA}(i)$  quantifies the level of “false-neighbour” point  $i$  suffer, because it is a sum of distance mismatch between the HD distance  $d_{ij}$  and the 2D distance  $\hat{d}_{ij}$  only when  $\hat{d}_{ij}$  is smaller than  $\sigma$ ; Similarly  $S_{NLM}(i)$  quantifies the pressure of “tear-up”. The local neighbourhood is defined by a global threshold  $\sigma$ , which is distance based. It also means that CheckViz assumes the scale of 2D and HD distance are the same, thus it only works for MDS family algorithms.

[Lespinats and Aupetit \(2011\)](#) give several arguments for choosing to colour the Voronoi cells instead of the points directly. However the shapes and sizes of the Voronoi cells are not consistent, because they depend on the densities of the points. Though the Voronoi diagram fills the background of the scatterplot efficiently, it also destroys the natural structure of gaps in the scatterplot. Nevertheless using Voronoi diagram is a matter of taste, more importantly there are three major weakness of CheckViz that I believe are the most harmful to its utility:

Firstly, The definitions of the two local measures only restrict CheckViz to MDS family methods but also assume the scale of the 2D and HD distance are the same, which is unfit to newly developed DR methods.

Secondly, the global distance-radius based threshold  $\sigma$  for both HD and 2D space is not as flexible as Venna's Precision and Recall.

Lastly, CheckViz is not intuitive to read, because even the colours of the cells tell the distortion scale each points suffer from the tear-up and false-neighbour, we still can not know which pair of points are incorporate visualised.

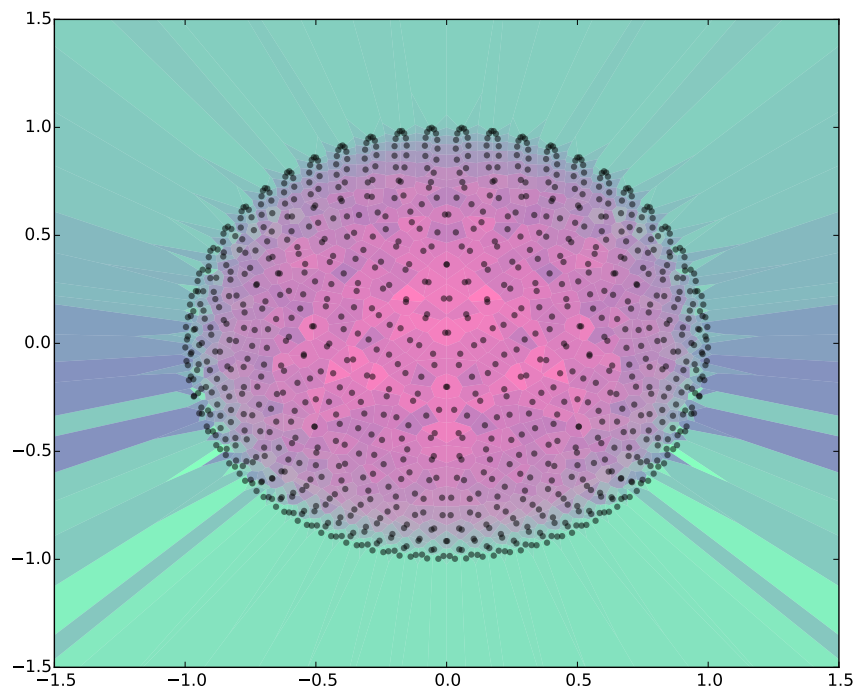
### 3.5.1 CheckViz of PCA Visualisation on 3D Sphere

This section presents the CheckViz plot for the PCA visualisation on the 3D Sphere only, as we learn that the scatterplot from PCA, MDS, NLM, LLE and Isomap are almost identical in the Chapter 2. CheckViz can not be applied to t-SNE visualisation directly due to the scale difference between the HD and 2D distances. CheckViz requires a global distance-radius based threshold  $\sigma$ . The original manuscript [Lespinsats and Aupetit \(2011\)](#) suggests to use the average distance between each data point to its fifth nearest neighbour as default. The colour scale in CheckViz is sensitive to  $\sigma$ . Since the 3D sphere is a very regular distributed dataset, the default parameter is good enough for this example.

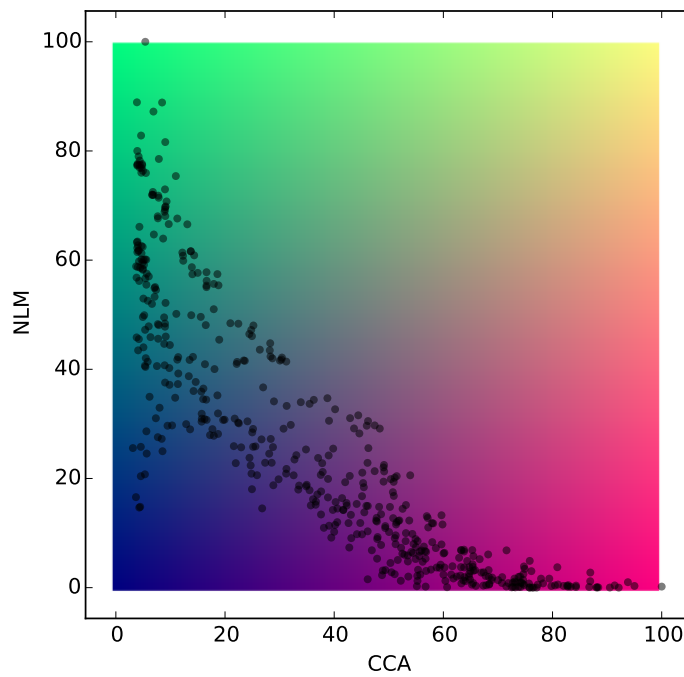
Fig. 3.5b is the 2D colour map of the CheckViz overlaid with the scatterplot of  $S_{CCA}(i)$  against  $S_{NLM}(i)$  for all the points. There are four parts in the colour table:

- The bottom-left corner (low in both  $S_{CCA}$  and  $S_{NLM}$ ) is the colour for the points with the least distortions.
- The bottom-right (high in  $S_{CCA}$  but low  $S_{NLM}$ ) is the colour for false-neighbour distortion.
- The top-left (low in  $S_{CCA}$  but high  $S_{NLM}$ ) is the colour for the tear-up distortion.
- The top-right (high in both  $S_{CCA}$  and  $S_{NLM}$ ) is the colour for the points suffer both tear-up and false-neighbour seriously, which is called *shuffled*.

The CheckViz of the PCA visualisation is shown in Fig. 3.5a with the global distance threshold being the average distance to the fifth nearest neighbour. The majority of the cells in the middle of the disk are in the colour of the extreme false-neighbours; While the outskirts tend to be the colour of high tear-up. It is hard to identify the exact position of the cell colour in the 2D colour map, because our perception to the colour of an object is affected by its background. The scatterplot overlaid on the colour map in Fig. 3.5b gives the exact distribution of the two quality measures for each points, from which we can learn that the PCA visualisation suffers both false-neighbour and tear-up heavily.



(a) Threshold  $\sigma$  Is The Average Distances to The Fifth Nearest Neighbour.



(b) Colour Map Overlaid with Points Based on The Two Local Measures.

Fig. 3.5 CheckViz of PCA Visualisation of The 3D Sphere.

### 3.6 Neighbour Plot

For each pairwise relationship  $(i, j)$ , there is always a HD distance value  $d_{ij}$  and a HD distance rank  $r_{ij}$  associated with it. If one regards the HD data scatterplot as a node graph, naturally the pairwise relationships are the edges in this graph. It is a rather simple idea but surprisingly I haven't seen any such research until [Bushati et al. \(2011\)](#). The Neighbour Plot shows  $k$ -nearest neighbour relationships for each point as arrows in the scatterplot visualisation of HD data [Bushati et al. \(2011\)](#). It is the only kind of diagram currently in the HD data visualisation literature that it is capable of displaying which point is truly close to which in the scatterplot. Some design details of the neighbour plot are as follows:

- The nearest-neighbour relationship is asymmetric. Thus it is drawn as an arrow instead of an edge. For example, the first nearest neighbour  $j$  of the point  $i$  is drawn as an arrow from point  $i$  to point  $j$ .
- The arrows are semi-transparent so that overlapping ones can be seen in the visualisation.
- The width of the very short arrows are deliberately wider so that they can be seen despite of their lengths.
- The HD distance values are used to colour the arrows in “jet” colour map by default. Hence the arrows with short HD distance appear in red and long ones are in blue.

The significant benefit of the neighbour plot is that it is an intuitive presentation of the centre problem of the HD data scatterplot. The HD similarities and 2D proximities among the data points are organised based on the scatterplot naturally. Typically the default parameter for neighbour plot is  $k = 2$ . There are four kinds of arrows for any pair  $(i, j)$  presented in the neighbour plot:

- **The short red arrow** indicates point  $j$  is very close to point  $i$  in the HD space and their proximity in the scatterplot suggests so.
- **The long red arrow** indicates point  $j$  is supposed to be very close to point  $i$  in the HD space but the pair is visualised far apart in the scatterplot, which is a tear-up error.
- **The short blue arrow** suggests the  $k$ -nearest neighbour point  $j$  of the point  $i$  is not relatively close in the HD space but they are put closely regardless.
- **The long blue arrows** suggests the same in the HD space as above and the pair of points are placed far apart accordingly.

The neighbour plot has the minimal assumption about the data but the diagram can be cluttered easily. It can show the tear-up distortions effortlessly but with many redundant visual elements. Also the neighbour plot can not show false-neighbours error by design. Despite all the weakness, the neighbour plot provides a new direction for the HD data visualisation by transforming the scatterplot into a node-link graph.

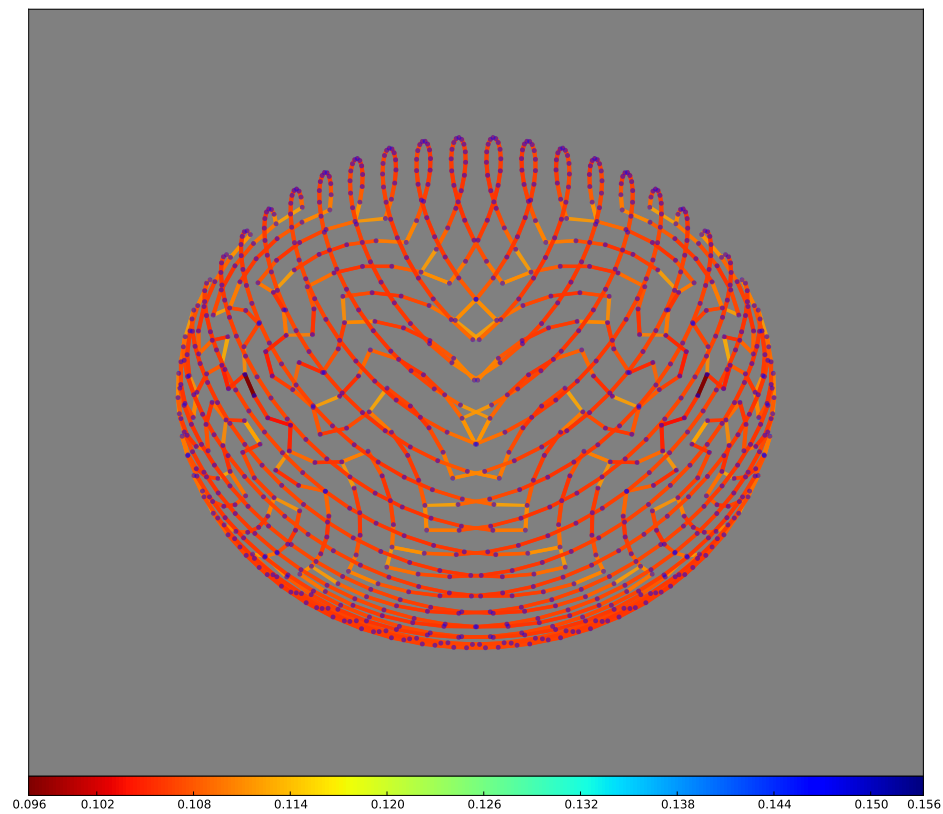
### 3.6.1 The Neighbour Plot of the 3D Sphere Visualisation

This section presents the neighbour plots of the PCA and t-SNE visualisations on the 3D sphere dataset mentioned in Chapter 2. We set the parameter  $k = 2$  for the neighbour plot and use the linear scale for colour mapping (shown in Fig. 3.6).

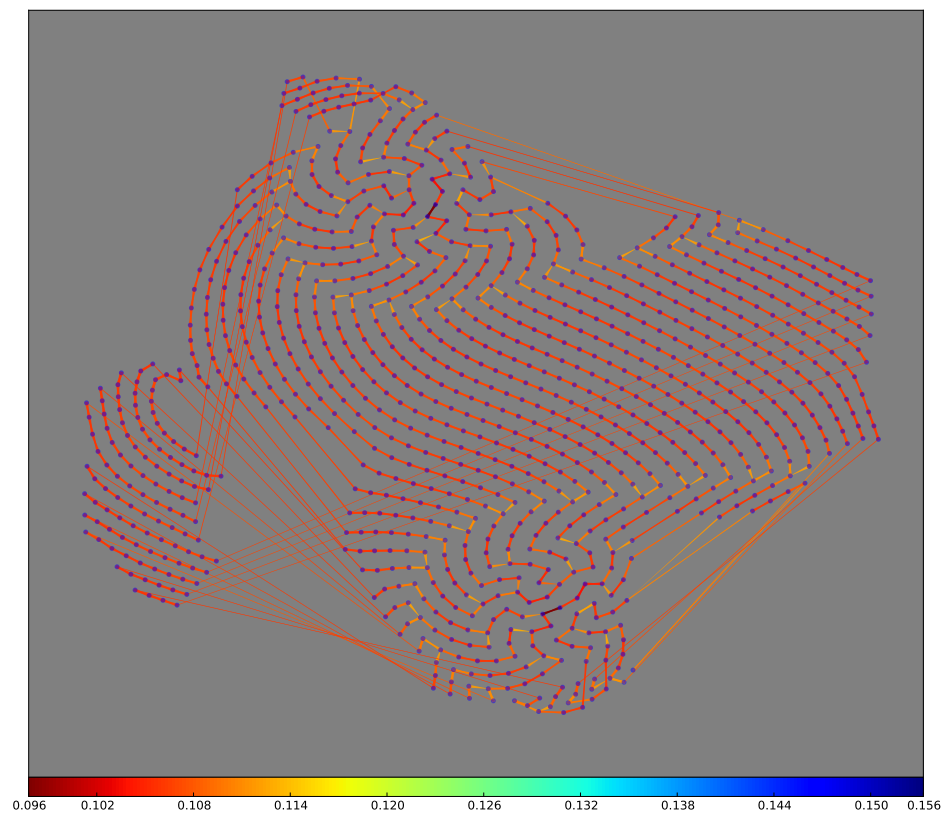
Because the 3D Sphere data items are distributed regularly, all the arrows in the both neighbour plots are in warm colours. Fig. 3.6a tells us PCA visualisation does not involve serious tear-up error because there are no red long arrows but many are crossing each other. Fig. 3.6b suggests that t-SNE seemingly splits the surface of the 3D sphere since there are some red long arrows connecting the edges of the patches. At this point, we shall agree that unlike the global visualisation quality measures or CheckViz, the neighbour plot can not only tell the presents of the tear-ups but also where they are in the scatterplot.

## 3.7 Conclusion

Knowing the global or local quality of a scatterplot of the HD data is essential but not enough, as we do not know where how the HD data visualisation are distorted. To my knowledge, Neighbour Plot is the first kind of the diagram that presents the visualisation distortion intuitively. However with the neighbour plot alone, we can never fully trust how good the local proximities are preserved in the scatterplot, because it can not reveal false-neighbours. It has a simple but crude design, but there are many aspects remain to be polished. The neighbour plot was the start point of my research and its design and usage are greatly improved which will be introduced in Chapter 5.



(a) PCA



(b) t-SNE perplexity = 30

Fig. 3.6 Neighbour Plots ( $k = 2$ ) of The 3D Sphere Visualisation.

# Chapter 4

## Delaunay Plot

### 4.1 Overlay Graph

Visualising high dimensional data by dimensional reduction methods can involve many unexpected distortions due to the fundamental difficulty of embedding the HD data into the 2D space. The underlying concept of the two arch kinds of distortion is based on the mismatch between the pairwise similarities in the HD space and the proximities in the 2D space. It is the entry point of the current global visualisation quality measurements. Meanwhile CheckViz [Lepinats and Aupetit \(2011\)](#) and Neighbour Plot [Bushati et al. \(2011\)](#) had pointed out another research direction, I believe there is no reason just to visualise the HD data as a plain scatterplot, while other visual elements like **lines** and **shade** can be added. I categorise such visualisation augmentation methods simply as *Overlay Graphs*.

The most obvious choice of the visual component of an overlay graph is the **edge** (or an arrow with the consideration of direction). Because the pairwise relationship between any two points is at the central of the problem and it can be represented by an edge essentially. Generally any kind of relationship between point  $i$  and point  $j$  (or from point  $i$  to point  $j$ ) can be written as a mapping function  $f(i, j)$ . An example of such in the context of the HD data scatterplot is the pairwise HD distance  $d_{ij}$  and HD distance rank  $r_{ij}$ .

Drawing pairwise relationships directly on the scatterplot transforms it into a node-link diagram (or graph for simplicity), in which the data points are the nodes and the relationships are the links with their colours encoded by the associated values such as their HD distances. Given  $N$  points, there are  $N \cdot (N - 1) / 2$  HD distances in total. Such a complete graph contains all information about how the data in HD space are distributed, but it is impossible to read for humans. Therefore the selection and visual design of the edges matter so that the balance between information and readability can be achieved.

This chapter will cover one of my original contributions to the overlay graph — Delaunay Plot, in which I will explain its motivation, edge selection scheme, the visual design and objective measures related to it.

## 4.2 Motivation of Delaunay Plot

The Neighbour plot is an overlay graph, because it selects the first- $k$  nearest neighbours of each point, which is a graph of  $N \cdot k$  arrows out of  $N \cdot (N - 1)$  in a complete graph. As discussed in section. 3.6, the arrows in the neighbour plot can reveal tear-ups, but the graph can never reveal any false-neighbours.

Instead of selecting the first- $k$  nearest neighbours of each point in the HD space, a natural step would be to select the first- $k$  nearest neighbours of each point in the 2D space and colour the arrows in HD similarities, which can certainly be able to select close neighbours in 2D space and reveal false-neighbours if there are any. In principle this method works, but the arrows in this kind of overlay graph can overlap and crossing with each other easily, which is not only unaesthetic but also hard to read.

There is no short answer to the question of how many close neighbours in the 2D space are need in order to reveal false-neighbour effectively. We propose to use a subset of all the edges from a Delaunay Triangulation [Delaunay \(1934\)](#) of the scatterplot as the selection scheme for the new overlay graph instead of the first- $k$  nearest neighbours in the 2D space. As with Neighbour Plot, the triangulation edges are coloured in HD similarity values so that those edges with relatively large HD similarity values can indicate false-neighbours. We term this kind of overlay graph as *Delaunay Plot*.

## 4.3 Geometric Properties of Delaunay Triangulation

Let  $Y = \{y_1, y_2, \dots, y_n\}$  be a set of two-dimensional points. A triangulation of  $Y$  is a maximal planar subdivision with points in  $Y$ . The Delaunay Triangulation (DT) is a maximal triangulation of  $Y$  such that the the circumcircle of any triangle in the triangulation does not contain a point of  $Y$  in its interior [De Berg et al. \(2000\)](#). However discussing Delaunay Triangulation in geometric detail is beyond the scope of this thesis. The section highlights some of its properties to provide an argument that the selection scheme is suitable for revealing false-neighbours.

- DT is the dual graph of the Voronoi Tessellation of given  $Y$ . Voronoi tessellation partitions the points into adjacent cells, while a DT edge is a straight line crossing the boundary of



the any two adjacent cells. So roughly speaking, the Delaunay edges select the “nearby” points for every point.

- Euclidean minimum spanning tree (MST) is a sub-graph of DT. and the nearest neighbour graph is a sub-graph of MST. Thus DT is guaranteed to include the nearest neighbour relationships for each point in the 2D space.
- The edges in DT are guaranteed not to cross with each other. This is the most important reason of favouring Delaunay triangulation as the selection scheme, as this elegant property will ensure a better diagram readability over the first- $k$  nearest neighbours.
- DT is a well studied topic in geometry. There are fast and stable algorithms to compute it [Lee and Schachter \(1980\)](#).

## 4.4 Primitive Delaunay Plot

Let  $E$  be the set of edges of the Delaunay Triangulation of a given two dimensional point set  $Y$ , and let  $(i, j)$  be a typical element of  $E$ . The primitive Delaunay plot is essentially to draw all Delaunay edges on top of the scatterplot and colour those edges according to their corresponding HD similarity values. The purpose of demonstrating the primitive Delaunay plot at first is to show what it can reveal and what remains to be polished.

Let us consider a synthetic data of a solid twin cubes in the 3D space. Each has 3 points on its edge, hence 27 points as a cube. I project this 3D dataset in the 2D space by MDS (using Euclidean distance) as shown in Fig. 4.1. As this the dataset is fairly simple, MDS visualises it reasonably well with two cubes separated apart and flattened in the 2D space. But as each cube (cluster) is solid, the 2D embedding must have some distortions within the cluster.

Fig. 4.2 is the primitive Delaunay plot of the solid twin cubes scatterplot. Each edge is coloured based on their corresponding HD distance values with a sequential colour map called *spring*. In the diagram, the short yellow edges indicate the pairs are “close” in the 3D space. The short “pink” and “orange” edges suggest the pairs have relatively bigger 3D distance values, thus these pairs are false-neighbours. In my view, the Delaunay edge is a better indicator for false-neighbour than the a Voronoi cell in CheckViz [Lepinat and Aupetit \(2011\)](#), because false-neighbours are reveal explicitly as edges instead of a pair of abstract numbers on a 2D colour table.

However HD distance is an exact but not intuitive HD similarity metric. In fact, if we enlarge the scale of one cube in the solid twin cubes drastically, the Delaunay edges of the other cube will certainly become all “yellow”, while the edges within the enlarged cube become all

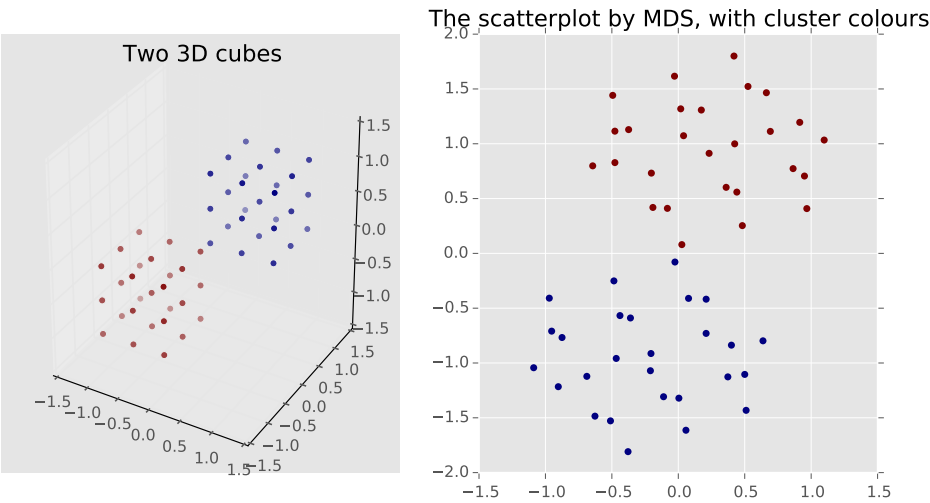


Fig. 4.1 The Solid Twin Cubes Dataset

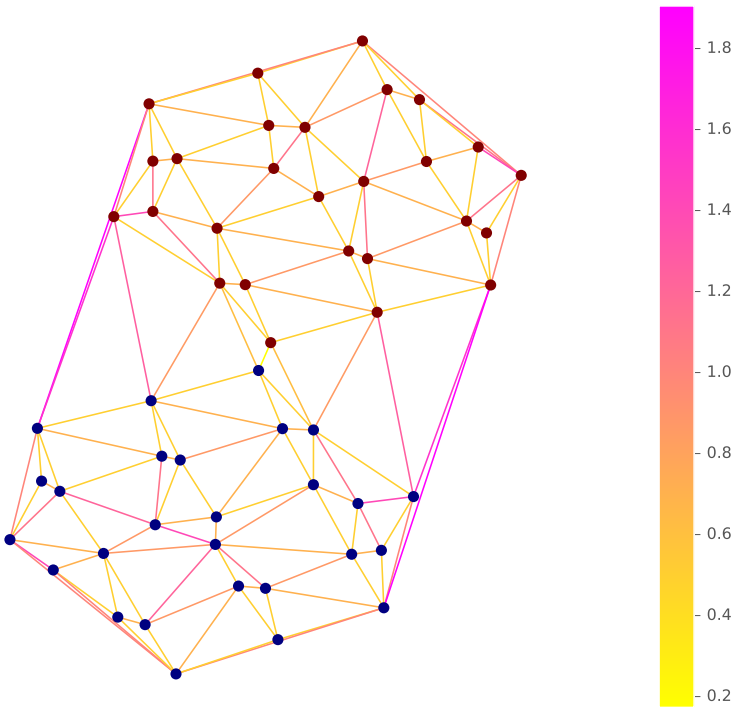


Fig. 4.2 Primitive Delaunay Plot of MDS Visualisation to the Solid Twin Cubes.

“pink”, which does not reflect the underlying patterns. A quick and useful way to fix this is to colour Delaunay edges based on their corresponding HD distance ranks instead. But high distance ranks do not always indicate “dissimilar” due to local density. Given the assumption that the viewer have very few prior knowledge of the data, we need a technique to *estimate* whether two points are “similar” or not compared to how other points are distributed locally in the HD space. In section. 4.5.4 I will introduce a new technique for such estimation. But be that as it may, the primitive Delaunay Plot is already capable of showing false-neighbour errors.

In addition, Delaunay edges generally select “nearby” pairs for each point in the scatterplot, only with the first nearest neighbours guaranteed, not the others. Drawing first- $k$  nearest neighbour arrows for each point is accurate in this context, but using Delaunay edges is a compromise between readability and accuracy. The  $k$ -nearest neighbour relationships are *asymmetric*, which should be drawn as *arrows*, while Delaunay edges can only indicate *symmetric* relationships. Nevertheless Fig. 4.3 is the histogram of  $k$ -nearest neighbour relationships in Fig. 4.2 selected by Delaunay plot, from which we can see how many  $k$ -nearest neighbour relationships are selected and missed. Since Delaunay edges are symmetric, one such edge  $(i, j)$  should contribute to the asymmetric 2D  $k$ -nearest neighbour relationship  $\hat{r}_{i,j}$  and  $\hat{r}_{j,i}$ . The histogram shows that all the first and second nearest neighbour relationships are selected, with a few missing of the third and fourth nearest neighbours. The number of hits decreases as the 2D distance rank increase.

However more importantly, there are a few high rank nearest neighbour relationships such as 20th, 30th and 40th are selected by the Delaunay edges. These Delaunay edges are not ideal because I want to restrict the purpose of Delaunay plot to probe “false-neighbours”. These high rank neighbour relationships are the result of **long** Delaunay edges. The two points of the pair highlighted by an undesired long Delaunay edge are already visualised far apart. If the pair is coloured as “pink”, which means they are “dissimilar” in the HD space, this is not useful because they are already far apart in the scatterplot. If the pair is “yellow” which means they are “similar”, then this is a “tear-up” error which can be better visualised by a neighbour plot, because its selection scheme is consistent to probe tear-ups. The next section introduces a technique to eliminate those long Delaunay edges.

The last problem with the primitive Delaunay plot is that a thin link is not an intuitive annotation for false-neighbour. In a node-link graph, “link” usually suggests the connected points are “closely related”, but the false-neighbour relationship has the opposite meaning. We need to change the design of the Delaunay edges.

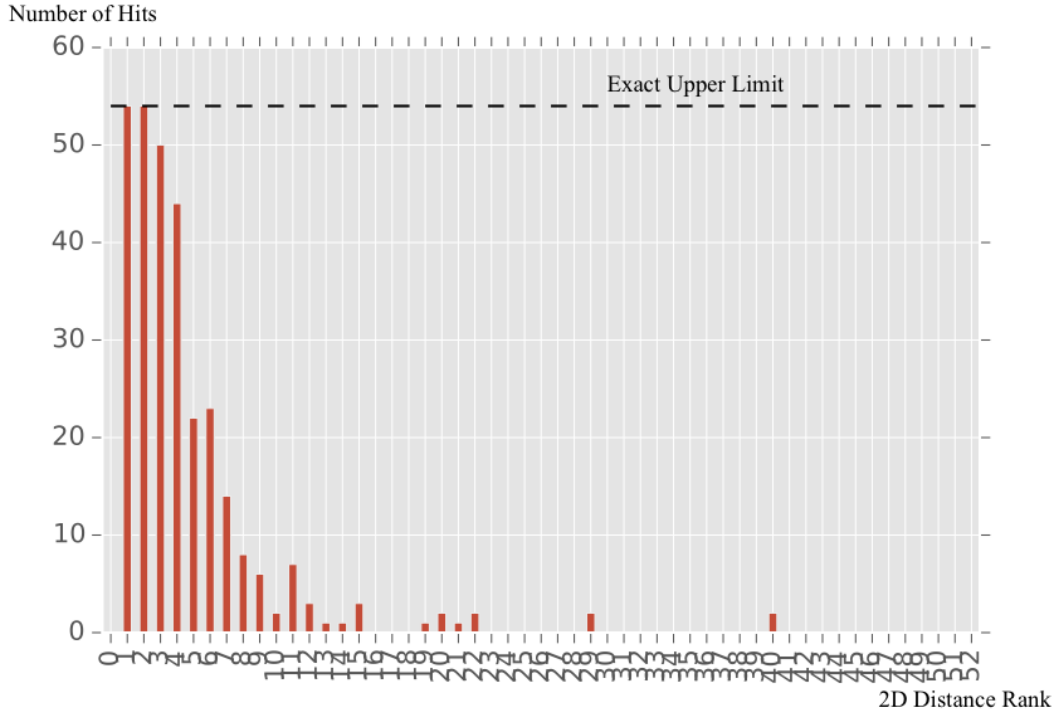


Fig. 4.3 Histogram of 2D  $k$ -nearest neighbour selected in Fig. 4.2

## 4.5 Refined Delaunay Plot

As the last section pointed out, there are three refinements needed:

- A cutting scheme for undesired long Delaunay edges;
- A suitable visual annotation for the Delaunay edge;
- An intuitive colour scale to reflect HD similarities based on local density for Delaunay Edges.

### 4.5.1 Cutting Long Delaunay edges

The basic utility of the Delaunay Plot is to show whether pairs of points are false neighbours or not, thus long Delaunay edges are useless and it must be trimmed from the Delaunay plot. Now the question is what is the quantitative definition of such Delaunay edges, since it is a rather vague and subjective term? Simply giving a threshold based on 2D distance and trimming all Delaunay edges longer than it would not work always, because in the 2D scatterplot, the densities of different parts vary but we expect the Delaunay plot to work for any scatterplot

produced by any DR method. Let us leave this question open and give the process a term as *Long Delaunay Edge Cutting Scheme*.

The initial quest is to select “nearby” pairs and draw them as edges, which Delaunay triangulation does it approximately with a few “undesired” exceptions, thus we need a subset of Delaunay edges. The key to the question is that the undesired long Delaunay edges are not suitable to represent “close” pairs compared with all other short Delaunay edges in a local sense. Thus local density estimation is needed. There could be a lot of ways to do so, the technique I propose here is highly inspired by SNE [Bunte et al. \(2012\)](#), which I called *Long Delaunay Edge Cutting Scheme via Stochastic Neighbour* (LDECS-SN).

### 4.5.2 Stochastic Neighbour

The first step of SNE translates the high dimensional data  $X$  into a conditional probability distribution  $P$  (see Eq. 2.7 in section. 2.7). The numerator of the conditional probability  $P_{j|i}$  of any two points  $i$  and  $j$  in respect of point  $i$  is the *affinity*  $p_{j,i} = \exp(-d_{ij}^2/2\sigma_i^2)$ , thus the Gaussian curve decreases drastically when  $d_{ij}$  is bigger than  $3\sigma_i$ . The local density centred any point  $i$  in the HD space varies, however SNE searches for a suitable  $\sigma_i$  to ensure the entropy of  $P_i$  to be exact as  $\log_2(\text{perplexity})$  as all other points, which is guaranteed to exist. This property ensures dense structure has lower  $\sigma_i$  than sparse structures.

Let the scalar  $s := d_{ij}/\sigma_i$ , it can serve as a global threshold to select close neighbours in the HD space. I term such method as *stochastic neighbour* and the scalar  $s$  as *stochastic neighbour score*. The parameter of this method is as the same as in SNE, which is *perplexity*. An advantage of stochastic neighbour (SN) is that the HD space neighbourhood is defined by an adaptable HD distance radius  $s \cdot \sigma_i$  but only controlled by a global threshold  $s$ . Like  $k$ -nearest neighbour or the fixed distance radius, stochastic neighbour is only an estimation, which can be useful when the user has little prior knowledge about the data.

#### LDECS-SN

Suppose we have an algorithm that is able to estimate local density of a set of data points and its interface is restricted to perform a binary query: is point  $j$  a close neighbour to point  $i$ . Notice that point  $i$  is a neighbour to point  $j$  is an asymmetric relationship. If two points both agreed with each other as their neighbours, the pair is a symmetric relationship as “nearby”. For each edge selected by a primitive Delaunay plot, LDECS tests whether a pair is a “nearby” relationship. The edge is excluded from the Delaunay plot if the pair falls the test.

Algorithm 1 LDECS-SN is a specific scheme of LDECS using Stochastic Neighbour as the local density estimator. Let  $Y$  be the scatterplot coordinates and  $E$  be the edge collection

of a primitive Delaunay plot. The query of stochastic neighbour is only *true* if the stochastic neighbour score  $s_{i,j}$  of a testing pair  $(i, j)$  is smaller than given threshold  $s$ .

---

**Algorithm 1** Long Delaunay Edge Cutting Scheme via Stochastic Neighbour
 

---

**Input:**  $E, Y, perplexity$  and  $s$   
**Require:** Init SN with  $Y$  and  $perplexity$   
**Require:**  $nE := \{\}$  {an empty collection of trimmed Delaunay edges}  
**Procedure**  
 $m = sizeof(E)$   
**for**  $i = 0$  **to**  $m$  **do**  
 $a, b \leftarrow E[i]$   
**if**  $SN.query(a, b, s)$  **and**  $SN.query(b, a, s)$  **then**  
append  $(a, b)$  to  $nE$   
**end if**  
**end for**  
**return**  $nE$

---

### LDECS-SN Parameter Settings Guideline

Using stochastic neighbour to determine a “nearby” relationship of a pair in  $Y$  require perplexity and the threshold  $s$ . The positioning of points in the 2D space is limited and the structure is flatten, so a small perplexity around 20 is reasonable. Once the conditional probability  $P$  is calculated from  $Y$  for specific perplexity setting, it does not need to calculate again. Then one could test the threshold  $s$  from 2.0 to 3.0 until undesired long Delaunay edges are excluded (usually 2.5 would give a desirable result).

### Example

Back to the Solid Twin Cubes example in section 4.4, Fig. 4.4 shows that LDECS-SN cuts off the undesired long Delaunay edges effectively. The two cubes join in the centre of the diagram only with a few relatively short Delaunay edges. In particular, Fig. 4.4a has fewer “long” Delaunay edges than Fig. 4.4b does. It is a subjective matter whether setting perplexity to 10 is better than 15. But the second one has more short Delaunay edges within the cubes which are important to reveal false-neighbours. Later I will introduce two visualisation qualities to measure the edge cutting objectively.

LDSCS-SN also works with more complex examples. Let us modify the Solid Twin Cube example a bit so that one cube is still in  $3*3*3$  size but with the scale enlarged (sparse cluster), while another cube is in  $4*4*4$  size but the scale is smaller; In addition an outlier is added (shown by Fig. 4.5). Fig. 4.6 demonstrates that LDECS-SN is capable if cutting undesired

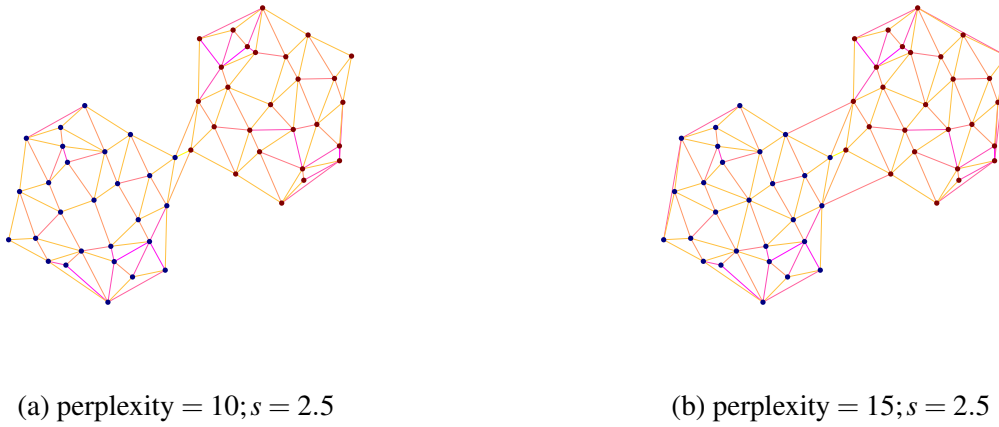


Fig. 4.4 LDECS-SN effects on Solid Twin Cubes Example

long Delaunay edges with the consideration of local density. Our perception of the outlier and clusters in the diagram are reinforced by such Delaunay Plot, while primitive Delaunay Plot destroys the natural gaps on the scatterplot.

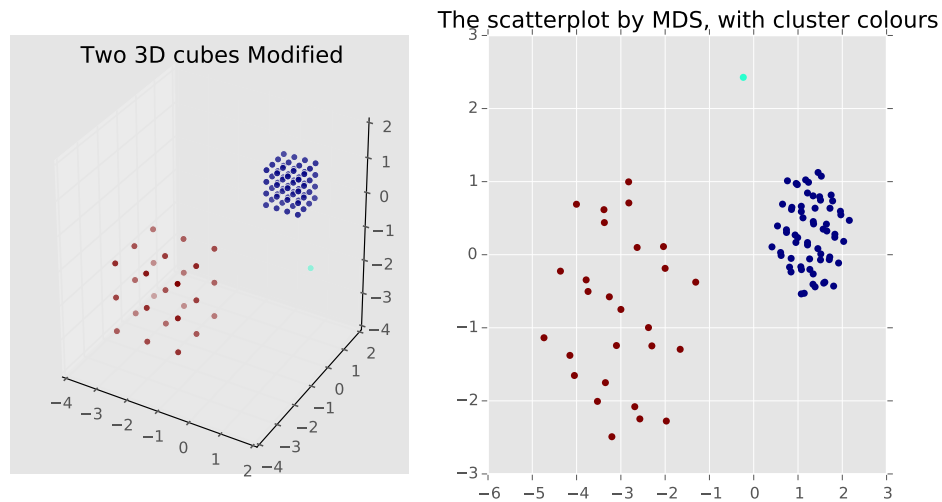


Fig. 4.5 Dense blue cube and sparse red cubs with one cyan outlier

### Limitation of LDECS-SN

Under the parameter guideline, LDECS-SN will no doubt trim the undesired long Delaunay edges from the primitive Delaunay plot, but not with geometric precision. In some situation, there can be unsatisfying cutting result like in Fig. 4.4b which the joint section between two

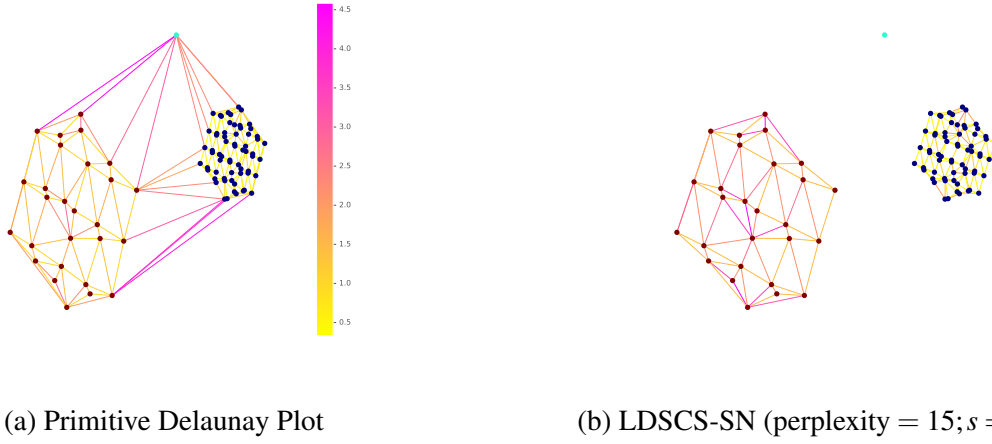


Fig. 4.6 LDECS Effects on The Modified Cubes Example

cubes have a few long edges but reducing perplexity or SN score could lead to fewer short edges within the cubes. In order to preserve more short edges and reveal false-neighbours, a trimmed Delaunay plot with only a few long edges is acceptable. The nature of the problem is subtle, I will leave the solution to Long Delaunay Edge Cutting Scheme open for future works.

### 4.5.3 New Visual Design of Delaunay Plot

In daily life, our minds are used to associating the “thin line” with “being related”, but the false-neighbours represented by the Delaunay edges has the exact opposite meaning — the pair are not similar in the HD space. Also the network of the thin-lines provides extra distraction to the viewer from observing the scatterplot. A Delaunay edge is a “link” in principle, but our solution is to draw it into a *thick, semi-transparent* and *blurred* rectangle connecting the pair. The considerations of the new Design are listed as follows:

- The Delaunay edges are guaranteed not to be crossing. Thus drawing the rectangles-shape edges instead of thin lines will not confuse viewers from observing the “nearby” relationships.
- The rectangle-shape edges are semi-transparent and blurred, filling the small gaps within the triangle simplex. This makes the Delaunay plot faded and faded into the background.
- Assuming LDECS-SN would cut the long edges well, the Delaunay plot becomes a mesh underneath the scatterplot, adjusting its extends based on the distribution of the points.
- The width of rectangle-shape edges is chosen as a fixed value. If there are dense clusters, the Delaunay edges within the clusters will be “melted” intensively and the meshes look



less transparent than the sparse clusters. Therefore, the local density information is reinforced visually by the opacity of the meshes.

Fig. 4.7 shows the new design of Delaunay Plot of the modified two cubes example projected by MDS. The pink Delaunay edges indicate possible “false-neighbour” relationships are annotated like a block instead of link. The gaps within each cube are filled but not the huge gaps between different cubes and outlier, due to LDECS. Although the edges are thick and blurred, the “nearby” relationships are still recognisable.

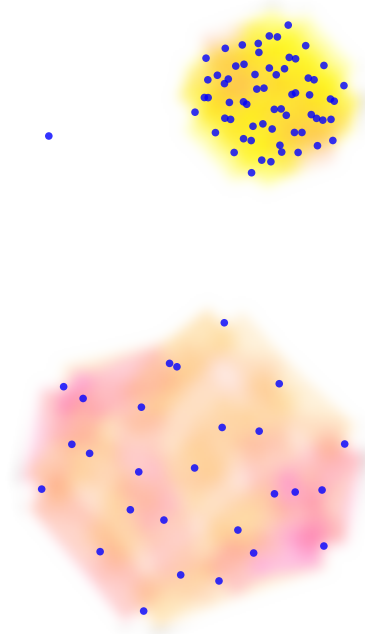


Fig. 4.7 New Design of the Delaunay Plot Example in Fig. 4.6

#### 4.5.4 Input Neighbourhood Estimation Scheme

Input Space Neighbourhood introduced in [Venna et al. \(2010\)](#), is a framework defining close neighbours centred at any point in the HD space based on thresholds of a pre-selected HD

similarity metric. The thresholds can be point-specific or absolute-global. The choice of a similarity metric and thresholds determines the prospective of “being closely related” that we interest. Apart from existing similarity metrics, Stochastic Neighbour (Section 4.5.2) can be an alternative prospective defining the input space neighbourhood, hence the stochastic neighbours score is regarded as a new similarity metric. For general purpose, I term the configuration of a similarity metric and thresholds for the input space neighbourhood as *Input Neighbourhood Estimation Scheme* (INES). Thus at this point there are primary similarity metrics mentioned in this thesis for INES: absolute distance (DIST), distance rank (kNN) and stochastic neighbour score (SN).

### Colour Scale for Coding INES

In principle, the colour of Delaunay edges can reflect any symmetric relationship  $f(i, j)$  for the pair  $(i, j)$ , for example in many cases, the absolute Euclidean distance. There are also a variety of interests for colouring the Delaunay edges with the average value of two asymmetric relationships, for instance, distance ranks, stochastic neighbour scores.

On the other hand, sometimes the continuous relationships values make the colour scale too diverse and difficult to probe false-neighbours (as discussed in Section 4.4). I propose to employ input space neighbourhood membership information to create a discrete colour scale for Delaunay edges. Suppose there is a preferred similarity metric and a (global) threshold, a Delaunay edge is marked as “close” if the two points linked by this edge are both in each other’s input space neighbourhood; if only one point is in the other’s neighbourhood, the edge is marked as “mediate”, otherwise the edge is marked as “faraway”. Therefore a Delaunay edge could be one of the three distinguish colours and the relationship it represented is transformed into symmetry.

An important instance of above undertaking is INES-SN-Discrete, which is a discrete colour scale for coding the Input Neighbourhood Estimation Scheme that uses Stochastic Neighbours. The process of creating the colour mapping table for Delaunay edges is presented in Algorithm 2, in which the function  $SN.query(i, j, w)$  is responsible to return a binary answer: is the stochastic score of point  $j$  in respect to point  $i$  lower than the global threshold  $w$ .

### INES-SN-Discrete Parameter Settings Guideline

INES-SN-Discrete requires two parameters: perplexity and the global SN score threshold  $w$ . The perplexity controls our basic assumption of the average number of relevant neighbours for each point in the HD space and  $w$  is responsible for testing the level of “being relevant neighbour”, typically ranging from 2.5 to 3.

**Algorithm 2** INES-SN-Discrete

---

**Input:**  $E$ ,  $X$ , *perplexity* and  $w$   
**Require:** Init SN with  $X$  and *perplexity*  
**Require:**  $estLst := \{\}$  {an empty list to hold boolean values of neighbourhood estimation for each Delaunay edge}  
**Procedure**  
 $m = sizeof(E)$   
**for**  $i = 0$  **to**  $m$  **do**  
     $a, b \leftarrow E[i]$   
     $q_1 \leftarrow SN.query(a, b, w)$   
     $q_2 \leftarrow SN.query(b, a, w)$   
    **if**  $q_1$  **and**  $q_2$  **then**  
        append “close” to  $estLst$   
    **else if**  $q_1$  **or**  $q_2$  **then**  
        append “mediate” to  $estLst$   
    **else**  
        append “faraway” to  $estLst$   
    **end if**  
**end for**  
**return**  $estLst$

---

**Example of Delaunay Plot Coloured in INES-SN-Discrete**

As we have some the prior knowledge of the Modified Two Cubes example in Fig. 4.5, every point in the cube is placed in 3D grid and there are 6 points equally close to the centre point of the cube, thus the setting for INES-SN-Discrete is *perplexity* = 6 and  $w = 3$ . Fig. 4.8 is the Delaunay plot of the modified two cubes example, applied with INES-SN-Discrete. In this diagram, the green edges indicate “close” relationship, the yellow ones indicate “mediate” and the red ones indicate “faraway”.

The estimation performance of INES-SN-Discrete is shown by Fig. 4.9. It is a histogram of all HD distance values selected by Delaunay Plot for each cube, in which bars are coloured with their INES-SN-Discrete estimation. Because every point in the cube is placed exactly on a 3D grid, the smallest pairwise HD distance is  $2/2 = 1.0$  in the red sparse cube and  $1/3 = 0.33$  in the blue dense cube. The histogram shows that the Delaunay edges labelled by INES-SN-Discrete with “close” are all the shortest HD distance. Thus the estimation is correlated with HD distance for each cube. It also proves that INES-SN-Discrete overcome the disadvantages of using HD Euclidean distance directly to colour the edges.

However there are some shortest pairwise HD distances that are labelled as “faraway” and “mediate” in blue dense cube. The situation is inevitable since INES-SN-Discrete is only an estimation of similarity. If we decide to increase the *perplexity* to 10, the size of input

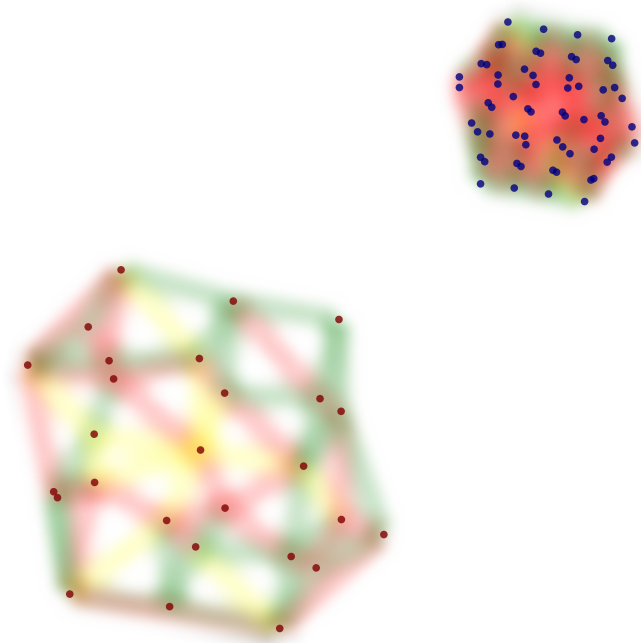


Fig. 4.8 Delaunay plot of the Modified Two Cubes Colour in INES-SN-Discrete (perplexity = 6)

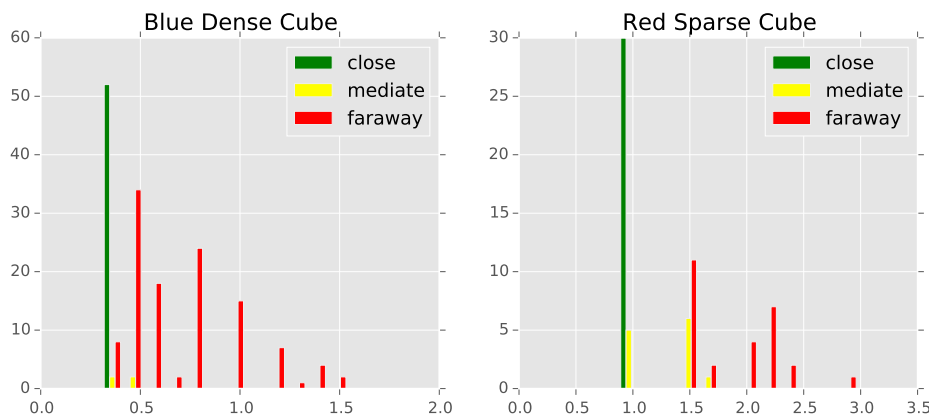


Fig. 4.9 Histogram of INES-SN-Discrete Performance to Fig. 4.8

neighbourhood will become slightly bigger. The histogram of INES-SN-Discrete’s performance shown in Fig. 4.11 is similar to Fig. 4.9.

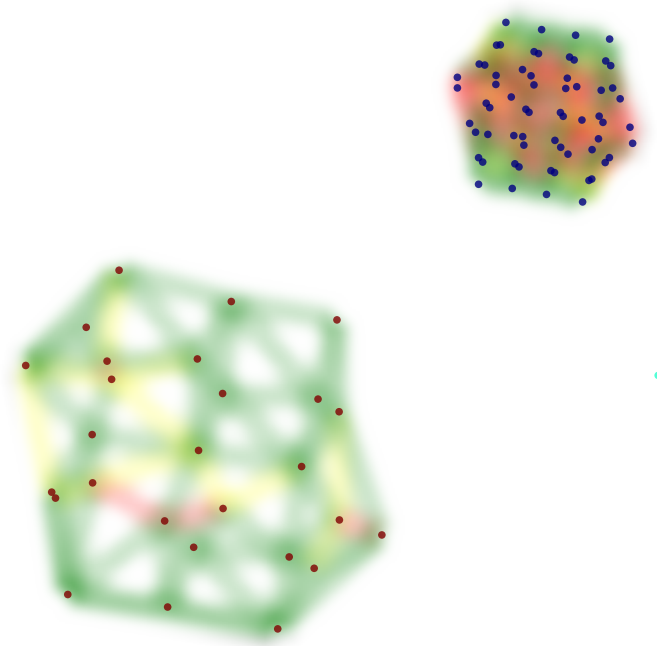


Fig. 4.10 Delaunay plot of The Modified Cubes Coloured in INES-SN-Discrete (perplexity = 10)

Lastly I will point out regardless the perplexity (6 or 10, even 15), the experiment shows that with  $w = 3$ , the size of input space neighbourhood cyan outlier is absolutely zero, which is a blessing. If the cyan outlier is wrongly placed inside either cube by a DR method, there will be red Delaunay edges drawn from it, marked as “faraway” by INES-SN-Discrete.

## 4.6 The Myth of Output Neighbourhood

Identifying visualisation distortions requires deciding the size of **both** input space neighbourhood and output space neighbourhood. These two kind of neighbourhood are equally important.

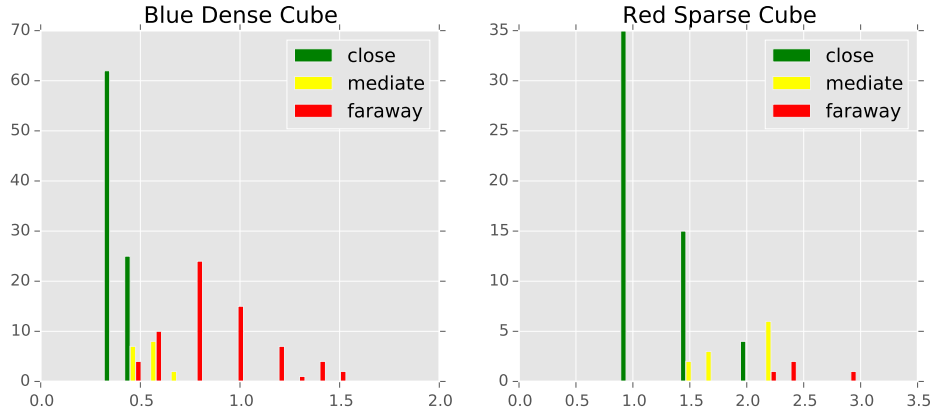


Fig. 4.11 Histogram of INES-SN-Discrete Performance to Fig. 4.10

*k*-Nearest neighbours is a widely used practice for defining the size of output neighbourhood and calculating a variety of visualisation quality measures such as Trustworthiness & Continuity Venna and Kaski (2001), Co-Ranking Histogram Lee et al. (2008), and Precision & Recall Venna et al. (2010), from which the visualisation performance of Dimensionality Reduction methods are indexed. Similarly a fixed 2D Euclidean distance radius  $r$  can also be used for defining the output neighbourhood. *k*-Nearest Neighbour (*k*NN) or fixed radius based method (DIST) is only a blind neighbourhood estimation in a blind HD space, but in the 2D space we can easily see their disadvantages. Firstly as the densities in different parts of the scatterplot vary, there won't be a global fixed *k*NN or DIST threshold suitable for selecting all reasonable neighbours in every possible situation. Secondly it is counter-intuitive for humans to track the size of neighbourhood around one single point in the scatterplot without any annotation, because the radius circle can not be drawn on every data point in the visualisation.

Those classical methods for defining the output neighbourhood are only designed for testing the capacity of DR methods not for human eyes. Output neighbourhoods that no human eyes can read are *invalid* for the purpose of visualisation, nor are the tear-ups and false-neighbours distortions they suggest. The size of output neighbourhood must take consideration of local density and our visual perception capability. Therefore It should be adapted and not too “big”. But how big it should be? What is the upper limit to this? And what is the better annotating method to suggest the size of output neighbourhood on the scatterplot?

#### 4.6.1 Defining Output Space Neighbourhood by Delaunay Neighbours

The refined Delaunay Plot (DP) is a node-link diagram with its edges assigned with labels of “close”, “mediate” and “far away”, in which one node could “travel” via Delaunay edges

to another if there is a route. Instead of trying to imagine an invisible radius circle in the scatterplot, we can track the path formed by connected Delaunay edges from one point to another.

For a single point  $i$  in the scatterplot, its Delaunay Neighbours (DN) are those points connected to it via a possible path in Delaunay Plot network. Furthermore the *first rank* Delaunay Neighbours of point  $i$  ( $DN(1)_i$ ) is a set of those points connecting to point  $i$  by the *shortest* paths in length of 1, in other words, they are directly connected; Similarly second rank Delaunay Neighbours of point  $i$  ( $DN(2)_i$ ) is a set of points connecting to point  $i$  by the shortest paths in length of up to 2. The upper limit of the rank of Delaunay neighbours of certain point  $i$  is the length of the shortest path to the furthest point it can travel to in the network of Delaunay Plot, denoted as  $DN(lim)_i$ .

There are three obvious advantages of using DN over KNN or DIST:

- Delaunay edges are visible to track. In the mesh of Delaunay Plot, it is not difficult for human to look up the a shortest path from one point to another. Even if Delaunay edges are blurred, the configuration of the Delaunay triangulation is friendly enough for humans to read for a small amount of points.
- Assuming *Long Delaunay Edge Cutting Scheme* could only select legitimate short edges, a small path length is a good approximation of being “close” in the 2D space.
- DN gives a adaptive output neighbourhood size but with an global and intuitive threshold for controlling the size of output neighbourhood, *the length of shortest path*.

To make Delaunay neighbour queryable, I propose to assign weights to Delaunay edges, which gives a symmetric adjacent graph. For a weighted undirected graph, Dijkstra’s algorithm [Dijkstra \(1959\)](#) can be applied. The most obvious method is to assign a weight of 1 to each edge in DP. Formally let  $G_{DN}$  be the DN query matrix for every pair relationship  $(i, j)$ ,  $E$  be the list of Delaunay edges in Delaunay Plot with each element to be a pair of indices of data point and  $n$  is the number of total data points. Algorithm 3 describes the above procedure.

## 4.7 Objective Measures of Delaunay Plot

The main purpose of Delaunay Plot is to show the worst kind of false-neighbours, because Delaunay edges select roughly all the “nearby” relationships in the scatterplot even without defining an extended output neighbourhood. When observing Delaunay Plot of the scatterplot visualisation, the most significant visual impact is the red edges embedded in the green meshes and we simply want to know how many red edges are there in DP.

**Algorithm 3** Delaunay Neighbour Query Matrix

---

**Input:**  $E, n$   
**Require:**  $G_{DN}$ , a matrix of  $n \cdot n$  size with all 0 values.  
**for**  $i \leftarrow 0, \text{sizeof}(E) - 1$  **do**  
     $i, j \leftarrow E[i]$   
     $G_{DN}[i, j] \leftarrow 1.0$   
     $G_{DN}[j, i] \leftarrow 1.0$   
**end for**  
 $G_{DN} \leftarrow \text{Dijkstra}(G_{DN})$   
**return**  $G_{DN}$

---

The first objective measure we suggest for a Delaunay Plot is *Cleanness*, which is the ratio between the number of Delaunay edges marked as “far away” and the total number of Delaunay edges in Delaunay Plot, the value is always between 0 and 1. Formally given input space neighbourhoods defined by any INES, any point  $i$  has an input space neighbourhood denoted as  $P_i$ . Let  $E_{DP}$  be the set of edges in Delaunay Plot, any edge  $(i, j)$  is considered as “far away” if  $i \notin P_j \wedge j \notin P_i$ . Such pair is also a “false-neighbour” because only “nearby” pairs are selected by Delaunay edges. The objective measure *Cleanness* is given in Eq. 4.1. The high the Cleanness is the fewer the worst kind of false-neighbours, the more likely the scatterplot visualisation is good.

$$M_{\text{Cleanness}} := 1 - \frac{|\{(i, j) | (i, j) \in E_{DP}, i \notin P_j \wedge j \notin P_i\}|}{|E_{DP}|} \quad (4.1)$$

However Delaunay edges can not select all “nearby” pairs in the scatterplot, thus if one missing “nearby” pair is false-neighbour, it can not be revealed by Delaunay Plot. Usually for given 2D layout its Delaunay Triangulation is unique, thus the number of such hidden false-neighbours are the innate limitation of Delaunay Plot. Fig. 4.12 is an example of Delaunay Plot missing a “nearby” pair. For the certain point  $i$ , its first rank Delaunay neighbours including all the nearby points except point  $j$ . If  $(i, j)$  is estimated as “far away”, then it is a false-neighbour distortion. But how to select such missing “nearby” pairs in the scatterplot?

Neither the first rank Delaunay neighbours nor k-Nearest neighbours can select all “nearby” relationship. It has to be distance radius circle and it has to be adapted. Finding missing “nearby” pairs in Delaunay Plot could be approximated by using the 2D distance from the certain point to its furthest first rank Delaunay neighbour, as we already accept using Delaunay edges to select nearby pairs previously. The new radius based output neighbourhood of point  $i$  is defined as  $Q'_i := \{j | \hat{d}_{ij} < \max_{k \in DN(1)_i} \hat{d}_{ik}\}$ , in which  $\hat{d}_{i,j}$  is the 2D distance in the scatterplot between point  $i$  and point  $j$ . Furthermore one has to notice that  $Q'_i$  gives an **asymmetric** relationship for a pair query  $(i, j)$ , because distance radius of  $Q'_i$  and  $Q'_j$  might be different. In case of missing



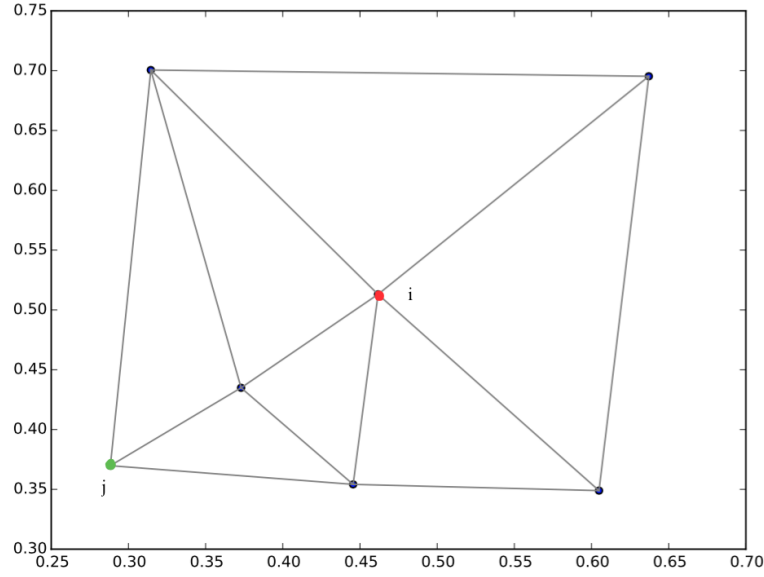


Fig. 4.12 An Example of Missing a “nearby” point (green) for central point (red) in Delaunay plot

out any “nearby” pair found by the new output neighbourhood, such pair is still considered as the worst kind of false-neighbour if its **symmetric** similarity relationship in HD space is “far away”.

The effectiveness of Delaunay Plot revealing the worst kind of false-neighbours is defined as a ratio between the number of the false-neighbours revealed by Delaunay edges and the number of false-neighbours found in the new radius based output neighbourhood. The second quality measure of a Delaunay Plot, the *Effectiveness* is defined as Eq. 4.2.

$$M_{\text{Effectiveness}} := \begin{cases} \frac{|\{(i,j) | (i,j) \in E_{DP}, i \notin P_j \wedge j \notin P_i\}|}{|L|} & |L| \neq 0 \\ 1 & |L| = 0 \end{cases} \quad (4.2)$$

$$L := \bigcup_{i=1}^n \{j | j \in Q'_i, j \notin P_i \wedge i \notin P_j\} \quad (4.3)$$

The closer  $M_{\text{Effectiveness}}$  approaches to 1 the less hidden false-neighbours are, and the more trustworthy the Delaunay Plot is in terms of revealing the worst kind of false-neighbours. Usually the numbers of revealed false-neighbours and hidden false-neighbours are positively correlated, so  $M_{\text{Cleanness}}$  is the **primary** quality measure of Delaunay Plot.

So far we have not been able to find a readable visual annotation to indicate hidden false-neighbours on Delaunay Plot. Be that as it may, like showing proximity relationships in Neighbour Plot [Bushati et al. \(2011\)](#), the pursuit of revealing all false-neighbours is endless. In this sense high dimensional data visualisation is always a visual perception puzzle where users accept estimated certainty and live with dark uncertainty, from which some truth of the beyond dimensions is learned.

## 4.8 The Implication of Transitive Proximity Relationship

Our eyes trick us and our minds take short-cuts, which is a reason why our visual perception sometimes is less reliable and why we are superior to machine in another respect. Delaunay Plot still needs to be read carefully. Suppose there is a 30-points tight cluster in the HD space isolated from all other structures, and they are visualised faithfully as a tight cluster in the 2D space. The input and output neighbourhood for certain point would not necessarily be the same size. For example the size of the input space neighbourhood of point  $i$  belong to this cluster is 29 but its output neighbourhood suggest by the second rank Delaunay neighbours has only 15 points. Thus it suffers 14 tear-ups in the visualisation. While in the Delaunay plot, all edges connecting these 30 points are marked as “close” in green colour. Our minds would take the short-cut by assuming the cluster is very like to be a faithful representation and expand the length of paths towards the upper limit. Such an interpretation is typical but not always correct.

Expanding Delaunay neighbours is a tempting and dangerous visual instinct for us, but sometimes also useful. Because we tend to think the proximity relationship is **transitive**, namely if there are visual evidences in Delaunay Plot suggest point  $a$  is close to point  $b$  and point  $b$  is close to point  $c$ , then point  $a$  is likely close to point  $c$  even they are not directly connected. To compensate our visual instinct with accuracy, I introduce the concept of *Delaunay Neighbour Reachable*.

### 4.8.1 Delaunay Neighbour Reachable

For any two points  $i$  and  $j$ , point  $j$  is Delaunay neighbour reachable (DNR) to point  $i$  if there exists a path formed **only** by Delaunay edges that are estimated as “close”. Delaunay neighbour reachable is still Delaunay neighbour relationship thus it is *symmetric* and it can be ranked. Similar to  $DN(k)_i$ ,  $DNR(k)_i$  means there exists a *shortest* path of at most  $k$  connecting Delaunay edges all estimated as “close”. For example point  $j$  is  $DNR(2)_i$  to point  $i$  if there exists a shortest path formed by up to 2 connecting Delaunay edges that are assigned with labels of “close” between them.

Like calculating the rank of Delaunay Neighbour using Dijkstra algorithm, weights need to be assigned to Delaunay edges to make DNR queryable. Particularly the Delaunay edges marked as “close” are assigned with weight of 1; “mediate” as  $n$ , where  $n$  is the total number of points in the scatterplot; “far away” as 0. The weighting scheme can guarantee the Delaunay edges in the shortest path connecting any two points that are Delaunay neighbour reachable are marked as “close” with only one kind of exception: there might exist a pair of points that are connected by one Delaunay edge not marked as “close”, but Dijkstra finds a DNR path another way around and overrides the label of the edge. The reason we should rule this exception out is that they are already “unreachable” (red colour edges), so the other Delaunay neighbour reachable route around is invalid.

Formally let  $G_{DNR}$  be the DNR query matrix for every pair relationship  $(i, j)$ ,  $E$  be the list of Delaunay edges in Delaunay Plot with each element to be a pair of indices of data point,  $n$  is the number of total data points and  $estLst$  be the estimation from INES for every Delaunay edge, such as Algorithm ?? . The procedure of obtaining Delaunay Neighbour Reachable Query Matrix is given in Algorithm 4.

---

**Algorithm 4** Delaunay Neighbour Reachable Query Matrix

---

**Input:**  $E, n, estLst$

**Require:**  $G_{DNR}$ , a matrix of  $n \cdot n$  size with all 0 values.

```

for  $i \leftarrow 0, \text{sizeof}(E) - 1$  do
   $i, j \leftarrow E[i]$ 
  if  $estLst[i] = \text{CLOSE}$  then
     $G_{DNR}[i, j] \leftarrow 1$ 
     $G_{DNR}[j, i] \leftarrow 1$ 
  else if  $estLst[i] = \text{MEDIATE}$  then
     $G_{DNR}[i, j] \leftarrow n$ 
     $G_{DNR}[j, i] \leftarrow n$ 
  else
     $G_{DNR}[i, j] \leftarrow 0$ 
     $G_{DNR}[j, i] \leftarrow 0$ 
  end if
end for
 $G_{DNR} \leftarrow \text{Dijkstra}(G_{DNR})$ 
for  $i \leftarrow 0, \text{sizeof}(estLst) - 1$  do
  if  $estLst[i] \neq \text{CLOSE}$  then
     $G_{DNR}[i, j] \leftarrow 0$ 
     $G_{DNR}[j, i] \leftarrow 0$ 
  end if
end for
return  $G_{DNR}$ 

```

---

By following only the shortest path of Delaunay edges marked as “close”, there would be a mismatch between DN and DNR query for pair  $(i, j)$ . For any pair that is Delaunay neighbour reachable, the rank of DN will not be greater than the rank of DNR, except Delaunay edge connected false-neighbours. The rank of DNR also indicates proximity reachability in the respect of human eyes, namely if  $G_{DNR}[i, j]$  is 0, then  $(i, j)$  is “far away” either they are “false neighbour” revealed by Delaunay edge (DN rank is 1) or there is no such shortest path connecting them with labels of only “close”. If  $G_{DNR}[i, j]$  is greater than  $n$ , it means the shortest path between point  $i$  and  $j$  has at least one Delaunay edge marked as “mediate”. Thus for a given Delaunay neighbours set  $DNR(k)_i$  of point  $i$  and  $k < n$ , the rank of DN of the Delaunay neighbour point  $j$  inside it could be smaller than  $k$ . In this sense, DNR expands the output neighbourhood of DN with no obvious false-neighbour included (the first rank DN of point  $i$  that are false-neighbours), which still could be tracked by human eyes.

#### 4.8.2 Noise in Delaunay Neighbour Reachable

Delaunay neighbour reachable is defined to serve our assumption that proximity relationship could be transitive. A point  $i$  travels on the path found by Algorithm 4 and reaches point  $j$ , these two points could be “close” in the HD space or not. The proximity relationship in the HD space of pair  $(i, j)$  is related two asymmetric input neighbourhood set  $P(k)_i$  and  $P(k)_j$ . The symmetric proximity relationship of  $(i, j)$  is “close” in the HD space, if  $j \in P(k)_i \wedge i \in P(k)_j$ , which is known from querying input space neighbourhood. The pairs that are Delaunay neighbour reachable **and** “close” in the HD space are called *true Delaunay neighbour reachable* (true DNR); While if they are Delaunay neighbour reachable and not “close” in the HD space, they are called *false Delaunay neighbour reachable* (false DNR).

In the situation the user looking at the Delaunay Plot and following the green path widely to estimate the proximity in the HD space, he/she might gets some unexpected false DNRs. The risk is presented in the form of Signal Noise Ratio, called Transitive Cleanness ( $M_{\text{TransitiveCleanness}}$ ), by treating all the true DNR pairs as signals and all the false DNR as noises. Therefore given the shortest path length value  $\hat{k}$ , the signal noise ratio is defined as,

$$SNR_{DNR(\hat{k})} = \frac{\text{number of true DNRs}}{\text{number of false DNRs}} \quad (4.4)$$

The number of true DNR and false DNR both increase with  $\hat{k}$ . Usually SNR value decreases as  $\hat{k}$  increases. The dramatic plummet in  $SNR_{DNR(\hat{k})}$  is a sign of great risk and uncertainty if we try to estimate the proximity relationship by tracking the green paths in the Delaunay Plot excessively.

However it is not that helpful by just giving users a global warning in the form of SNR and discouraging them from looking at the green paths. In some parts of the scatterplot the HD proximity relationships are indeed transitive and those structures are very important. But It is difficult to annotate true DNR and false DNR on Delaunay Plot. So what is a better graphical representation for discovering true Delaunay neighbour reachable and escaping from the constraint of Delaunay Neighbours?

### 4.8.3 Faithful Delaunay Patches

Transitive proximity relationship means one point is true Delaunay neighbour reachable to another and it is possible to identify such path. But highlighting such path can only be used to show one pair of points not the others. It is also possible for a group of points to have transitive proximity relationships, which means they are **mutually** true Delaunay neighbour reachable in Delaunay Plot. The latter one is a more interesting and stronger constrained question. All the points in such a group are “close” to each other in the HD space and the paths between any two of them are all green. However it has three problems: first it is computationally expensive because of the possible combinations of the group setting; Second, the points in such group are only the end points of the green paths not including the middle points; third even the group is found, the design of visualisation has to highlight both the end points and their paths. Such a visualisation can only show one such group once otherwise the paths might overlap while one edge can only be given one colour to indicate which path it belongs to.

The Delaunay plot is an incomplete Delaunay Triangulation, due to some long Delaunay edges are cut. It possesses sufficient triangle simplexes especially in cluster structures. Instead of inquiring the green paths between pairs, we can shift our focus on one Delaunay triangle and ask are these three points in the triangle “close” to each other? Furthermore given the points in one Delaunay triangle are *mutually* true DNR, we can move on and ask its adjacent triangles that are they mutual true DNR within themselves and which one of them is also mutual true DNR to the given triangle? This problem has a even stronger constraint than mutual true DNR within a group of points. It leads to a segmentation of Delaunay triangles such that all the points within the triangle group are mutual true DNRs. I call such a group of adjacent Delaunay triangles a *Faithful Delaunay Patch* (FDP).

Since some long Delaunay edges are cut, the triangles with missing edges are excluded from searching. Also the triangle that is not self-mutual-true-DNR is filtered out. Now for each triangle, there could be at most three adjacent triangles in Delaunay Plot. By treating all Delaunay triangles as nodes, the Delaunay triangulation is transformed into a graph  $T$ . The goal of the algorithm is to obtain a sub-graph  $G_{tri}$  of  $T$  such that each connected component in  $G_{tri}$  is a faithful Delaunay patch. The algorithm performs breadth-first-search (BFS) from one

random node in  $T$  at start. During visiting one node, only its adjacent nodes that has not ever been in BFS queue (not been checked) can be test for triangle-wise mutual-true-DNR asserting, which means every vertex in the adjacent node (actually just one new point) has to be true DNR to every vertex of every nodes in the already connected component in  $G_{tri}$ . If the assertion is true, the adjacent node is added to the connected component and BFS queue. When the queue is empty, the algorithm will select another random node in  $T$  that is self-mutual-true-DNR and not been checked, then the similar process is applied again, until all nodes in  $T$  have been checked. The detail pseudo-code is given as Algorithm 5, in which  $G_{DNR}$  is the Delaunay neighbour reachable query matrix,  $G_{DN}$  is the Delaunay neighbour query matrix and  $G_P$  is the symmetric proximity relationship query matrix ( $G_P[i, j]$  is true when  $j \in P(k)_i \wedge i \in P(k)_j$ ) obtained from input neighbourhood estimation scheme.

Algorithm 5 is essentially a breadth first search applied to a sparse graph. Its pseudo-code has some lines using Object Oriented Programming paradigm.  $T$  is the object of Delaunay triangulation applied on a given 2D layout. It has two “methods”.  $T.triangles$  gives an indexable list of Delaunay triangles, in which each triangle has three vertices.  $T.adjacentTrianglesIndexof(i)$  gives up to 3 indices of Delaunay triangles in the list given by the former method that are adjacent to the triangle with its index in the list being  $i$ .  $G_{tri}$  is triangles against triangles symmetric matrix representing an adjacent graph. The algorithm gradually builds  $G_{tri}$  into a sub-graph, in which each connected component of nodes (triangles) is a faithful Delaunay patch.  $G_{tri}.connectedComponentOf(i)$  returns a list of triangle indices referring to  $T$  that are already connected in  $G_{tri}$ , which essentially is giving all positions in the row vector  $G_{tri}[i]$  where the elements are true. Lastly function `mutualTrueDNR(connectedTriangleIndices, newTriangleIndex, T,  $G_{DNR}$ ,  $G_P$ )` is an assertion testing whether the vertices of the new triangle are mutual-true-DNR to every vertex of all triangles that are already connected in  $G_{tri}$ . To judge mutual-true-DNR, the function needs  $G_{DNR}$  and  $G_P$  passed as parameters.

The segmentation of faithful Delaunay patches is not unique, but it has some nice and intuitive properties:

- All points in the patch are mutually true DNR to each other. By excluding all false DNRs, no uncertainty is evolved.
- There are no Delaunay edges estimated as “far away” (false-neighbour) or “mediate” in the patch.
- DFPs compensate our visual instinct and tell us where the true DNR could reach but with a very strong constraint.

- The patch is in fact a subset of the **intersection** of input neighbourhoods of all points in the patch. It is a sign of a tight cluster or manifold in the HD space, which is visually highlighted.
- Because of input neighbourhoods intersection, the total number of points in one patch is not greater than the size of the smallest input neighbourhood of such point in the patch.
- If two patches are adjacent to each other (with no gaps), they share some points on the boundary. It is a sign that the two patches are “close” to each other forming a bigger cluster or manifold in HD space, which is visually highlighted.

$G_{tri}$  is the abstract form of faithful Delaunay patches. Since each patch is a collection of adjacent Delaunay triangles, one can draw the patch in semi-transparent on the scatterplot with different colours to distinguish them. The diagram is an overlay graph, which I term as *Faithful Delaunay Patches Map*. It is possible to colour different patches in the map based on Four Colour Theorem [Gonthier \(2008\)](#) or Five Colour Theorem [Heawood \(1949\)](#). Unfortunately I have not yet implemented an efficient colouring algorithm. Currently the colouring strategy is to use a random function to generate RGB vectors for patches and avoiding pale colours like white. Although it is rather naive method, its practical performance suggests the chance of two different patches sharing the same colour is very small. The sophisticated colouring strategy is left for future work.

The faithful Delaunay patches map implements a very strong constraint by requiring at least some of the triangles being self-mutual-true DNRs. It is only worthwhile to draw such patch map based on the Delaunay Plot with high cleanness, otherwise there would be very few or no patches satisfying the condition. Moreover since the patch is a subset of intersection of input neighbourhoods for every point in it, the constraint rules out asymmetric input neighbourhood relationships leaving only tight structure. In order to expand the intersecting input neighbourhood or the size of the patch, it is not harmful to implement bold estimation by increasing the size of input neighbourhood just for the faithful Delaunay patches map alone. In this case k-Nearest neighbour is especially useful for giving such bold input space neighbourhood estimation due to its clear intuition and simplicity.

## 4.9 Examples of DP and FDPM

In the end of this chapter, we present some examples of Delaunay Plot and Faithful Delaunay Patches Map to demonstrate our methodology. The showcase datasets are: 3D Spherical Shell and 3D Gaussian Spherical Shell.

### Showcase: 3D Spherical Shell

The 3D spherical shell dataset is presented as the showcase dataset in Chapter 2 and 3. The perplexity of t-SNE for producing the scatterplot is 30. The input neighbourhood estimation scheme is Stochastic Neighbours with its perplexity = 20 and  $w = 2.5$ . The faithful Delaunay patches map uses the previous Delaunay Plot but using k-Nearest neighbours ( $k = 40$ ) as its input neighbourhood estimation scheme which helps faithful Delaunay patches to expand.

As discussed in chapter 3, t-SNE is the only DR method that excels in visualisation quality measure on the dataset of 3D spherical shell. Fig. 4.13 is the Delaunay plot of t-SNE scatterplot of the 3D spherical shell applied with INES-SN-Discrete (perplexity = 20,  $w = 2.5$ ). The Delaunay Plot Cleanliness is 0.991. The Diagram gives undeniable evidences that the visualisation preserves the local structures faithfully, as most of the Delaunay edges are green. There are only a few red edges shown due to the LDECS-SN does not cut them, which are not harmful. But without faithful Delaunay patches map we still don't know whether the points are just visualised correctly only with their first rank Delaunay neighbours.



Fig. 4.13 Delaunay Plot of t-SNE Visualisation on The 3D Sphere



Fig. 4.14 is the faithful Delaunay patches map of t-SNE ( $perplexity = 30$ ) visualisation of spherical shell with input neighbourhood estimation scheme as KNN ( $k = 40$ ). The diagram shows the segmentation of the big green patch. many faithful Delaunay patches are bigger than basic Delaunay triangles which means the proximity relationships among the points inside the patch is transitive. Most of the faithful Delaunay patches are connected without gaps, which means the patches share their boundary in almost every direction. This is a clear indication that the 3D sphere shell is split and flatten in 2D space.



Fig. 4.14 Faithful Delaunay Patches Map of t-SNE Visualisation on The 3D Sphere

### Showcase: 3D Gaussian Spherical Shell

The Gaussian spherical shell dataset with 1000 points in 3D space is similar to the previous spherical shell dataset but its points are randomly distributed on the 3D surface instead of being evenly distributed (Fig. 4.15a). The Gaussian sphere is less regular than the previous and it has some holes on the surface. The perplexity of t-SNE for producing the scatterplot is 30. The input neighbourhood estimation scheme is Stochastic Neighbours with its perplexity = 20 and

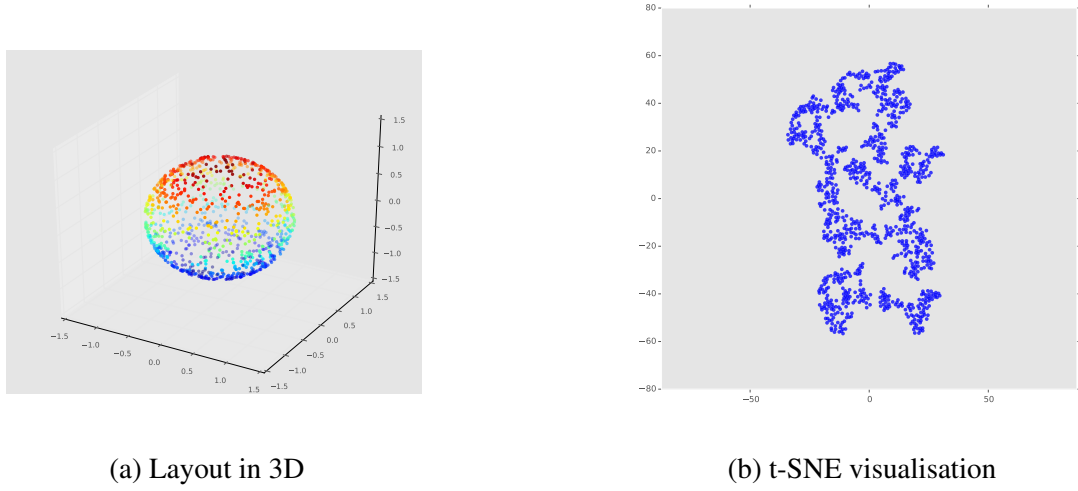


Fig. 4.15 3D Gaussian Spherical Shell Example

$w = 2.5$  (Fig. 4.15b). The faithful Delaunay patches map uses the previous Delaunay Plot but using  $k$ -Nearest neighbours ( $k = 40$ ) as its input neighbourhood estimation scheme.

The Delaunay plot helps us to inspect true neighbours and false neighbours of the nearby points connected by Delaunay edges in the visualisation. The input neighbourhood estimation scheme is given slightly smaller perplexity than the parameters of t-SNE because we expect the conservative assumption on the input neighbourhoods could let Delaunay edges to probe potential false-neighbours. Fig. 4.13 tells us that the visualisation of t-SNE is in good quality because most of the Delaunay edges are green and hence the cleanness is high. But Fig. 4.16 cannot tell us any transitive proximity properties of Delaunay edges, as there are big green patches across the diagram.

Faithful Delaunay patches map shown in Fig. 4.17 suggests the transitive properties of Delaunay edges in previous Delaunay plot. As the big green patches are segmented into smaller faithful Delaunay patches, the visualisation give clear evidences that the points within each faithful Delaunay patch are indeed mutually proximate to each other hence the Delaunay edges connecting them are transitive. Moreover, the faithful Delaunay patches are adjacent to each other in some places. It implies that those patches form a bigger patch in Gaussian spherical surface.



Fig. 4.16 Delaunay Plot of t-SNE Visualisation on Gaussian Sphere

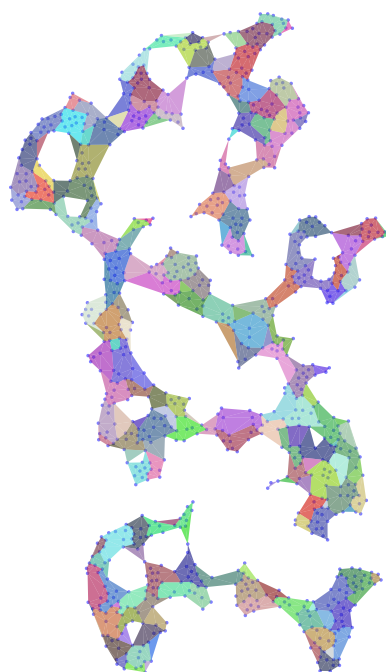


Fig. 4.17 Faithful Delaunay Patches Map of t-SNE visualisation of 3D Gaussian Sphere

## 4.10 Conclusion

At present when we talk about the Overlay Graph, an augmentation to the scatterplot of high dimensional data, it is about selecting a meaningful subset of relationships in the latent complete-graph of the high dimensional data, from which many possibilities are unfolded. The chapter introduced Delaunay Plot, a relationship subset selection using information already exposed in 2D space. Integrating with estimated knowledges of high dimensional data, Delaunay Plot helps to identify the worst false-neighbour, redefining the output neighbourhood and advising our visual instinct. At first it is a compromise overlay graph with consideration of both accuracy and readability, but later it proves that it has much potential with many possible progressive improvements and objective quality measures such as, Cleanness & Effectiveness, Delaunay neighbour, Delaunay neighbour reachable, Transitive Cleanness and Faithful Delaunay Patches. It is an example of just how much we can do about relationship selection and how much we can learn from high dimensional data using the overlay graph than the plain scatterplot visualisation.

---

**Algorithm 5** Finding Faithful Delaunay Patches
 

---

**Input:**  $T, G_{DNR}, G_{DN}, G_P$   
**Require:**  $n \leftarrow \text{sizeof}(T.\text{triangles})$   
**Require:**  $k \leftarrow$  the number of all points in the scatterplot  
**Require:**  $\text{mask} \leftarrow \{\text{false}, \dots, \text{false}\}$  in the size of  $n$   
**Require:**  $G_{tri} \leftarrow \text{matrix}()$  in size of  $(n, n)$ , all **false**, except the diagonal being all **true**  
**Require:**  $\text{queue} = \{\}$   
 for  $i \leftarrow 0, n-1$  do  
    $a, b, c \leftarrow T.\text{triangles}[i]$   
    $m \leftarrow k > G_{DNR}[a, b] > 0$  and  $G_P[a, b]$   
    $m \leftarrow m$  and  $(k > G_{DNR}[b, c] > 0$  and  $G_P[b, c])$   
    $m \leftarrow m$  and  $(k > G_{DNR}[a, c] > 0$  and  $G_P[a, c])$   
    $m \leftarrow m$  and  $(G_{DN}[a, b] = 1$  and  $G_{DN}[b, c] = 1$  and  $G_{DN}[a, c] = 1)$   
    $\text{mask}[i] \leftarrow m$   
 end for  
**Require:**  $\text{checked} \leftarrow \neg \text{mask}$   
    $\text{startIndex} \leftarrow$  a random position in  $\text{mask}$  that is not **false**  
    $\text{queue.append}(\text{startIndex})$   
    $\text{checked}[\text{startIndex}] \leftarrow \text{true}$   
 while **true** do  
   while  $\text{queue}$  is not empty do  
      $i \leftarrow \text{queue.pop}()$   
      $\text{triangle} \leftarrow T.\text{triangles}[i]$   
     for  $j \in T.\text{adjacentTrianglesIndexOf}(i)$  do  
       if  $\neg \text{checked}[i]$  and  $\text{mask}[j]$  then  
          $m \leftarrow \text{mutualTrueDNR}(G_{tri}.\text{connectedComponentOf}(i), j, T, G_{DNR}, G_P)$   
         if  $m = \text{true}$  then  
           for  $t \in G_{tri}.\text{connectedComponentOf}(i)$  do  
              $G_{tri}[t, j] \leftarrow \text{true}; G_{tri}[j, t] \leftarrow \text{true}$   
           end for  
            $\text{queue.append}(j)$   
            $\text{checked}[j] \leftarrow \text{true}$   
         end if  
       end if  
     end for  
   end while  
   if  $\text{checked}$  is not all **true** then  
      $\text{startIndex} \leftarrow$  a random position in  $\text{checked}$  that is **false**  
      $\text{queue} \leftarrow \{\text{startIndex}\}$   
      $\text{checked}[\text{startIndex}] \leftarrow \text{true}$   
   else  
     break  
   end if  
 end while  
 return  $G_{tri}$

---

# Chapter 5

## Proxigram

### 5.1 Overview

Reviewed in section. 3.6 Neighbour Plot [Bushati et al. \(2011\)](#) is the only overlay graph that shows the HD proximity (similarity) relationships directly on the scatterplot. It is an intuitive, simple but primitive overlay graph. In this chapter the idea of Neighbour Plot is refined and it is given a new name as *Proxigram*.

Proxigram is essentially a meaningful subset of the latent complete-graph in high dimensional data, dedicated to display the proximity or similarity relationships. In other words, it shows which point is “truly close” to which in the scatterplot visualisation. This chapter presents how Neighbour Plot is generalised as Proxigram, why global control for the level of truth is necessary, what is the strategy to reduce visual clutter and preserve crucial proximity relationships?

### 5.2 From Neighbour Plot to Proxigram

#### 5.2.1 Definition

Given a collection of high dimensional data  $X$ , for any point  $i$  its input space neighbourhood  $P_i$  is obtained by an input neighbour estimation. The *General Proxigram* is an overlay graph showing the all proximity relationships for every point in  $X$ . In this sense Proxigram holds our assumption of the neighbourhoods in high dimensional space. Formally the definition of General Proxigram is given in Eq. 5.1.

$$\text{Prx}_{\text{general}} := \bigcup_{i=1}^n \{(i, j) | j \in P_i\} \quad (5.1)$$

In this thesis there are three input neighbourhood estimation scheme: HD Euclidean distance radius, k-nearest neighbours and stochastic neighbours, resulting in three kinds of Proxigram: Proxigram- $r$ , Proxigram- $k$  and Proxigram- $\sigma$  respectively. The proximity relationship  $(i, j)$  in  $\text{Prx}_{\text{general}}$  is not symmetric unless HD distance is used as the input neighbourhood estimation scheme. Therefore unlike Delaunay Plot, the relationships are drawn as arrows instead of edges in Proxigram, which is called *proxi-arrow*. There could be other formations of Proxigram if other input neighbourhood estimation scheme is employed.

Generally the applicability of Proxigram- $r$  is very limited, because high dimensional Euclidean distance become less contrast with increasing dimensionality and even in the low dimensionality, like 3 or 4, the disadvantage of using Euclidean distance to define input neighbourhood is obvious. k-nearest neighbours is a more favourable choice. But it ignores any local density information so that the outliers has the same number of proxi-arrows as others. It makes the outlier in Proxigram totally unrecognisable even it is visualised correctly in the scatterplot. Stochastic neighbours is considered as a reliable and robust input neighbourhood estimation scheme, although it is less intuitive and the size of neighbourhoods varies.

### 5.2.2 Visual Appearance

The visual appearance of proxi-arrows in General Proxigram is slightly different from Neighbour Plot. Apart from being a polygon shape and semi-transparent arrow, the colour of the new proxi-arrow is not restricted to reflect Euclidean distance of the HD data. Apart from dedicating the colour scale for HD similarity or proximity metric value, the members within an input space neighbourhood can be ranked, the characteristic of which I term as the *level of proximity*.

The level of proximity in either k-nearest neighbours or stochastic neighbours is in fact ordered based on high dimensional Euclidean distance values. To make level of proximity intuitive to read, I propose to use the nature HD distance rank within an input neighbourhood for the alternative proxi-arrow colour scale. Also it is useful to indicate the size of neighbourhood globally, thus the upper limit of colour map index is corresponding to the largest HD distance rank in the input neighbourhoods of all points.

Mapping HD distance rank  $k_{i,j}$  into colour map index  $I$  requires normalisation. Eq. 5.2 defines two normalisation functions. The first one  $I_1$  is linear and in most cases it should be fine. The second one  $I_2$  emphasis on low distance ranks and the scale for larger ranks becomes shrink and less contrast, which is useful when the size of a few input neighbourhoods are relatively too large and dwarfing the colour scale of others. Usually “jet” colour map is used, which in particular, the normalised index value 0 is for red and 1 for blue.



$$I_1 := \frac{1}{K} \cdot k_{i,j} \quad (5.2)$$

$$I_2 := 1 - \frac{1}{2e^{k_{i,j}}/K} \quad (5.3)$$

$$\text{where } K := \max_{i=1}^n |P_i| \quad (5.4)$$

### 5.2.3 Visual Cues in General Proxigram

General Proxigram does not deviate from Neighbour Plot too much but it requires a sensible choice of input neighbourhood estimation, which we should treat as the temperate “ground truth” of HD similarities. For a reasonable input neighbourhood estimation, true outliers in HD data shall have no proxi-arrows and no other points have proxi-arrows pointing to them. This is why I suggest to use stochastic neighbours to replace k-nearest neighbours as the estimation scheme. While Neighbour Plot typically displays two nearest neighbours for every data point, whether the point is an outlier, belongs to sparse cluster or dense cluster is left unexplained. When inspecting the proxi-arrow individually, relatively short proxi-arrows indicate the pair of points are visualised legitimately, otherwise it signals a **tear-up** distortion, which is the kind of distortion Delaunay Plot can not reveal in principle.

When there is a sufficient amount of points placed “closely” to each other in the scatterplot and they are truly “close” in HD space, a natural consequence is that there are many short proxi-arrows condensing within this area. In this sense General Proxigram exposes clusters in a coarse way. The visual impact of this is huge. But such a “feature” can also be interpreted negatively because it is redundant, since the points are already close to each other in the 2D space. Furthermore, those short arrows in the area are overlapping and covering the scatterplot too much such that if there are some points that do not belong to this cluster (false-neighbours) the viewers would not be able to tell which points are responsible for those reaching out tear-up proxi-arrows. On the other hand, such points responsible for reaching out tear-up proxi-arrows could also have proxi-arrows pointing to the data points inside the cluster. The former situation has a totally different meaning from the latter one, but if the short arrows cover the scatterplot too much no one could be able to see. However without a sufficient level of proximity to display, there might not be such proxi-arrows at all.

The tear-up proxi-arrows are relatively **long** and those with high level of proximity are most important signals from Proxigram. But unfortunately they are likely crossing and covering the visualisation. It is expected that the appearance of Proxigram can be really a mess when the 2D data map does not preserve the proximity relationships properly. To avoid this, I suggest to

draw Delaunay Plot at first and then judge if Proxigram is worth to draw based on the *Cleanness* measure.

But in fact General Proxigram is already hard to observe even when only a small proportion of the points are misplaced. For example when drawing Proxigram- $k$  ( $k = 5$ ) on the scatterplot of 1000 points with 100 of them misplaced, there could be  $100 * 5$  tear-up proxi-arrows crossing and  $900 * 5$  short proxi-arrows covering the scatterplot visualisation.

Thus given a scatterplot visualisation with its Delaunay Plot *Cleanness* reasonably good, General Proxigram can still be frustrating to track the proximity relationships. It only takes the information from the HD space without utilising any information from the 2D space, leaving the cluttered diagram for our eyes. Later the chapter, I present three strategies to mitigate the visual clutters: a control variable for the level of truth, showing only tear-up proxi-arrows and showing mutual proximity relationships as minimum spanning forest.

#### 5.2.4 Examples of General Proxigram

General Proxigram draws all proximity relationships of input neighbourhood estimation  $P_i$  for every point  $i$ . The amount of proxi-arrows to draw is  $\sum_{i=1}^n |P_i|$ . The general Proxigram is expected to be cluttered even for a conservative input neighbourhood estimation. Here I present two examples of general Proxigram for the dataset of 3D Gaussian spherical shell shown in Fig. 4.15a.

Fig. 5.1 is the general Proxigram of the t-SNE visualisation (*perplexity* = 30) of the 3D Gaussian sphere with its input neighbourhood estimation scheme using kNN ( $k = 4$ ). Therefore there are  $4 \times 1000$  proxi-arrows to draw, which is a large amount but clearly merely 4-nearest neighbour can hardly be considered to be a suitable input neighbourhood estimation for the dataset. However the choice of the estimation scheme is valid because the dataset has no outliers.

The short proxi-arrows are indeed the majority in Fig. 5.1. The colour map used is “jet” and the normalisation function for mapping HD distance rank is linear (see Eq. 5.2). Thus the first nearest neighbour is red, but they are almost invisible to see, because they are very short compared to other proxi-arrows. This means that the scatterplot preserves nearest neighbour proximity relationships very well. Blue proxi-arrows are the fourth nearest neighbours relationships. A few of them are relatively long, which indicates that roughly from the fourth nearest neighbour, the tear-up distortion begins to occur in the scatterplot.

Fig. 5.2 is the general Proxigram of the t-SNE (*perplexity* = 30) visualisation of the 3D Gaussian sphere with its input neighbourhood estimation scheme using SN *perplexity* = 20,  $w = 2.0$ . There are 14073 proxi-arrows in the diagram, so about 14 neighbours for each point in average. It is a sensible input neighbourhood estimation as we already know the

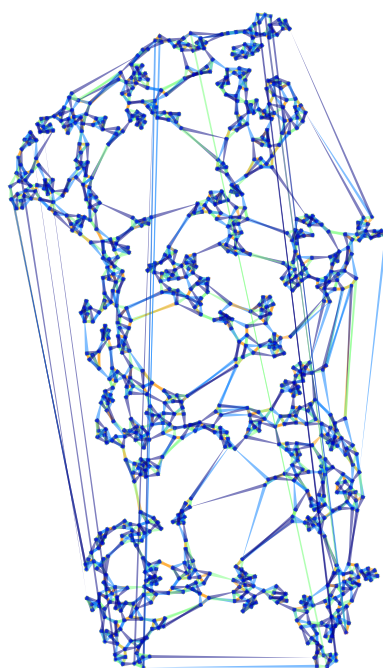


Fig. 5.1 General Proxigram of the 3D Gaussian Sphere, INES-KNN

distribution of the dataset in 3D. Although the diagram is cluttered, we can still tell short proxi-arrows are the majority in the diagram, which means the local structures are preserved very well. Notice that there are some red proxi-arrows, showing different patterns from Fig. 5.1. This is because HD distance rank like 3 or 4 is coloured as red in this diagram as the average size of  $P_i$  is 14. It has become clear that the main problem of Fig. 5.2 is that there are sufficient tear-up distortions shown but we can not see the actual source and target points of these tear-up, as they are covered by the short proxi-arrows.

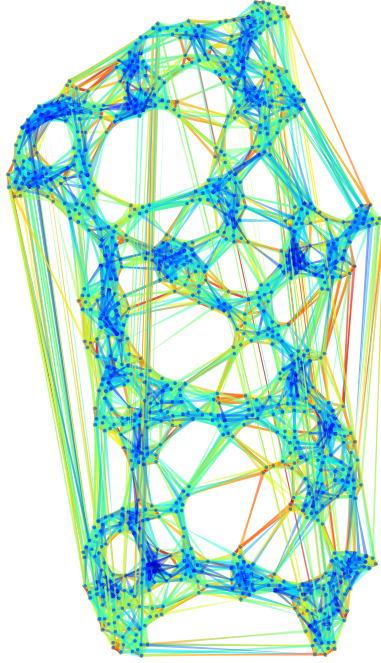


Fig. 5.2 General Proxigram of the 3D Gaussian sphere, INES-SN

### 5.3 Level of Truth

Stochastic neighbours seems to be a natural input neighbourhood estimation scheme. But the size of input neighbourhood for each point can be undesirably large or even exaggerated. In each input neighbourhood, the level of proximity decrease as the rank of HD distance order goes up. To compromise accuracy for readability, I introduce the concept, *Level of Truth*.

In each input neighbourhood  $P_i$  of point  $i$ , there are  $|P_i|$  points estimated as neighbours to point  $i$ , ordering in high dimensional distance  $d_{i,j}$  from smallest to largest. The higher the level of proximity the more important the relationship is. Here I define a global threshold  $\lambda$  between 0 to 1 to control the size of input neighbourhoods in order to select only a few with relatively higher level of proximity. Such filtered Proxigram is called  $\lambda$ -Proxigram, the formal definition is given in Eq. 5.5, in which  $P_i^\lambda$  is the adjusted subset of  $P_i$ .

$$\text{Prx}_\lambda := \bigcup_{i=1}^n \{(i, j) | j \in P_i^\lambda\} \quad (5.5)$$

$$P_i^\lambda := \{j | j \in P_i \wedge r_{i,j} \leq \lceil |P_i| \cdot \lambda \rceil \} \quad (5.6)$$

$$\text{where } r_{i,j} := |k : d_{ix} < d_{ij} \vee (d_{ik} < d_{ij} \wedge k < j)| \quad (5.7)$$

If we accept  $P_i$  as the temperate ground truth, the level of truth variable  $\lambda$  divides the proxi-arrows in two categories: *present and absent*. Since when using stochastic neighbours or HD distance radius as the estimation scheme, the neighbourhood sizes are unknown. Therefore the level of truth in the filtered Proxigram in respect of control variable  $\lambda$  is defined as the ratio between the number of present proxi-arrows and the total number of proxi-arrows in original, formerly given in Eq. 5.8.

$$L(\lambda) = \frac{|\text{Prx}_\lambda|}{|\text{Prx}_{\text{general}}|} \quad (5.8)$$

Although Level of Truth in the Proxigram a trivial concept, it is important to maintain the logical separation between the parameters of input neighbourhood estimation scheme and the global Proxigram visibility control, because Delaunay Plot also requires input neighbourhood estimation to be consistent with Proxigram. Furthermore if the prior knowledge of input space neighbourhood is known, the input space neighbourhoods configuration shall remain constant, but with adjustable level of truth, we can gradually learn the HD data distribution.

Fig. 5.3 is the  $\lambda$ -Proxigram, which is a sub-graph of the general Proxigram shown in Fig. 5.2, with its level of truth factor  $\lambda = 0.8$ .  $\lambda$ -Proxigram only partially addresses the problem of general Proxigram of being too cluttered when using stochastic neighbour as the input neighbour estimation scheme.

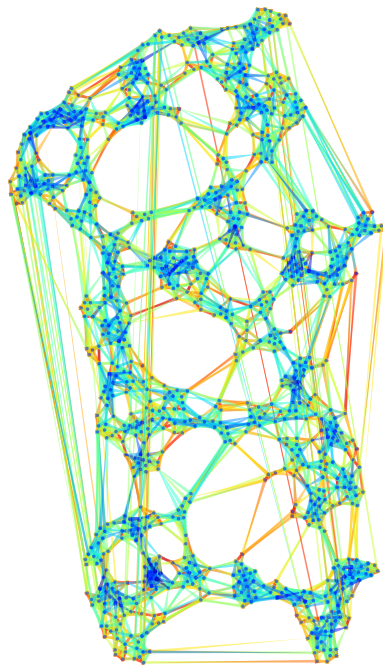


Fig. 5.3 The  $\lambda$ -Proxigram ( $\lambda = 0.8$ ) of the 3D Gaussian sphere, INES-SN

## 5.4 Tear-up Proxigram

### 5.4.1 Definition

General Proxigram or  $\lambda$ -Proxigram only take information from input neighbourhoods not utilise any information from output neighbourhoods. The short proxi-arrows are in some ways redundant and covering the scatterplot significantly. However Proxigram shows tear-up relationships, relatively long proxi-arrows and they are considered as more important. This section presents a new strategy of filtering proxi-arrows based on output neighbourhoods, called *Tearup-Proxigram*, which will reduce the number of proxi-arrows significantly.

Given an output neighbourhood  $Q_i$  for a typical point  $i$ , formerly Tearup-Proxigram is a subset of General or  $\lambda$ -Proxigram defined as Eq. 5.9.

$$\text{Prx}_{\text{tearup}}^{\lambda} = \bigcup_{i=1}^n \{(i, j) | j \in P_i^{\lambda} \wedge j \notin Q_i\} \quad (5.9)$$

Based on input neighbourhood estimation schemes and the methods of defining output neighbourhood, there are several combinations of tear-up relationship selection I would like to discuss. The notion is arranged as *Input neighbourhood Estimation Scheme vs Output Neighbourhood definition*:

### 5.4.2 Tear-up Filtering Scheme

**HD distance vs 2D distance** : The filtering scheme is actually based on the classical Shepard Plot [Shepard \(1962\)](#). It seems very promising that one can observe the Shepard Plot and select the left upper entries as tear-ups. But I have to point out the readability of Shepard Plot is poor and the thresholds of HD and 2D distance radius are hard to find. Because as discussed before the global distance radius value is difficult to form valid input or output neighbourhood definitions. With the dimensionality of HD space increasing, the HD distance becomes less comparable with 2D distance. Thus the tear-ups the filter scheme selects would be very unlikely legitimate.

**kNN vs kNN** : The filtering scheme is based on Co-ranking histogram [Lee et al. \(2008\)](#) that one can observe and select the left upper corner entries from it. Observing Co-Ranking histogram could hardly select legitimate tear-ups unless the data is very regular and well-sampled. The reason is similar as Shepard Plot, the thresholds are hard to find and kNN ignores local density information.

**SN vs DN** : The filtering scheme is based on Delaunay neighbours. I **recommend** it because SN reflects local density information, allowing empty input neighbourhood for outlier and adaptable input neighbourhood size. Delaunay neighbours is easier for eye to track and also account local density in the 2D space.

**SN vs DNR** : The filtering scheme requires users to tolerant false Delaunay neighbour reachable but it expands the output neighbourhood. Thus more proximal arrows can be trimmed. The maximum tear-up filtering can be achieved via DNR is to use the upper limit of DNR and tolerate all possible false DNR reachable pairs. It can be useful to reveal the **worst** tear-up only. Another benefit is that the reachable path on Delaunay Plot has natural control of upper limits, the path will end when there is a gap on the mesh or no green route available, while neither kNN nor 2D distance radius has this feature.

### 5.4.3 Categories of Tear-up

Colours are precious for visualisation. The source point of an tear-up proximal arrow can be one of two categories: the one has some other proximal arrows pointing to its “nearby” points in  $\lambda$ -Proxigram and the one that doesn’t have. Especially in the former kind, the source point has non-empty intersection with **all** its “nearby” points can be seen as the “joint” point for two connecting structure, because the point is “close” to all its “nearby” points are the one that visualised correctly and if the point also has tear-up proximal arrows reaching out of the cluster it is a signal that this point is the joint. Thus for all the tear-up proximal arrow(s) emitted from it is called *Joint tear-up*. While another more general situation in the former kind, the source point is “close” to at least one but not all “nearby” points (intersection of input neighbourhood is not empty). The tear-up proximal arrow(s) emitted from such source point is called *Subtle tear-up*.

There are two scenarios in the latter kind. The source point has “nearby” points but none of them is “close” to it in the HD space (the intersection of input neighbourhood with it is empty). this kind of source point is a total “false-neighbour”. The tear-up proximal arrow(s) emitted from such source point is called *False-neighbour Tear-up*; While the remaining case is that the source point doesn’t have any “nearby” points according to its output space neighbourhood. Thus it is visualised as an outlier in the 2D space, but it has proximal arrows which means it is not outlier in the HD space. The tear-up proximal arrows emitted from such source point is called *False-outlier Tear-up*. The difference between the four kinds of tear-ups are important to annotate in Proxigram. Therefore I suggest the colour scheme in Tearup-Proxigram should take advantage of the information from output neighbourhood, like in Delaunay Plot.

In order to inspect the input neighbourhood intersection between the source point  $i$  and its “nearby” points  $j$ , I used the temperate output neighbourhood  $Q'_i$  which is defined by the



2D distance radius of the furthest first rank Delaunay neighbour of point  $i$  (see section. 4.7). The precise definitions of those four kinds of tear-up are given in Eq. 5.10, each as a subset of tear-up proxi-arrows.

$$T_{\text{joint}}^{\lambda} := \{(i, j) \in \text{Prx}_{\text{tearup}}^{\lambda} \mid \forall k \in Q'_i, P_i^{\lambda} \cap P_k^{\lambda} \neq \emptyset\} \quad (5.10)$$

$$T_{\text{subtle}}^{\lambda} := \{(i, j) \in \text{Prx}_{\text{tearup}}^{\lambda} \mid \exists k \in Q'_i, P_i^{\lambda} \cap P_k^{\lambda} \neq \emptyset\} \setminus T_{\text{joint}}^{\lambda} \quad (5.11)$$

$$T_{\text{false-outlier}}^{\lambda} := \{(i, j) \in \text{Prx}_{\text{tearup}}^{\lambda} \mid Q'_i = \emptyset\} \quad (5.12)$$

$$T_{\text{false-neighbour}}^{\lambda} := \{(i, j) \in \text{Prx}_{\text{tearup}}^{\lambda} \mid \forall k \in Q'_i, P_i^{\lambda} \cap P_k^{\lambda} = \emptyset\} \quad (5.13)$$

$$\text{where } Q'_i := \{j \mid \hat{d}_{ij} < \max_{k \in DN(1)_i} \hat{d}_{ik}\} \quad (5.14)$$

Finally I suggest to use four distinguishable colours for the four kinds of tear-up proxi-arrows separately.

#### 5.4.4 Proxi-Efficiency

Neither General Proxigram nor  $\lambda$ -Proxigram can help to define any visualisation quality measure because they do not build with any information from the already existing 2D scatterplot. Intuitively we might think the less tear-up proxi-arrows are the better the scatterplot visualisation is. But it is tricky because a scatterplot that smashes all points into one position does not have any tear-up proxi-arrows at all and this is also the worst possible case we want to avoid. Nevertheless less tear-ups is indeed a sign of good quality scatterplot. Here I define an objective quality index for Tearup-Proxigram called *Proxi-Efficiency*. It is a ratio between the selected proxi-arrows by tear-up filtering scheme and the original amount of proxi-arrows in  $\lambda$ -Proxigram. The higher Proxi-Efficiency means more short proxi-arrows and less tear-up arrows, which is only a necessary condition for a good scatterplot visualisation. The measure is defined in Eq. 5.15.

$$M_{\text{Efficiency}} := 1 - \frac{|\text{Prx}_{\text{tearup}}^{\lambda}|}{|\text{Prx}_{\lambda}|} \quad (5.15)$$

#### 5.4.5 Bundling the Tear-up Proxi-Arrows

Tear-up are the long proxi-arrows in principle. These arrows trend to cross each other and considerate amount of them can dominate the graph.

While some tear-up proxi-arrows have potential patterns because they are initiating and targeting to the points that are close together in the scatterplot. In order to reduce the clutter in

the visualisation and reveal high level tear-up proximal patterns, we suggest to employ edge bundling method Mingle [Gansner et al. \(2011\)](#) to the Tear-up Proxigram. The edge bundling algorithm and my technical contributions are discussed in detail in Appendix 1.

Edge bundling can only deal with edges not arrows. Thus when applying edge bundling to tearup-Proxigram, the shape of arrow is lost and if two proximal arrows are reversed to each other, they are treated as one. An edge bundling algorithm will group visually similar edges together and render them as splines, while the unbundled ones remain as straight-lines. Furthermore a bundle edge can not be coloured with asymmetric relationship value. Due to this nature, the colouring scheme for a bundled Tear-up Proxigram is not as sophisticated as previous overlay graph. For example, using a single colour for all bundled edges with the intensity from pale to intense for indicating the number of bundled edges from low to high. Finally due to edges are bundled, the source and target points of an proximal arrows are not explicitly indicated, but on the other hand, edge bundling only occurs when the endpoints of the edges are close together, so the compromise is acceptable.

#### 5.4.6 Example of Tearup-Proxigram

Fig. 5.4 is the tearup-Proxigram corresponding to the  $\lambda$ -Proxigram in Fig. 5.3, with its output neighbourhood defined by Delaunay neighbours (within 2 rank). Four distinguishable colour are used to indicate the type of tear-ups, joint – pink, subtle – grey, false-outlier – sky blue and false-neighbour – purple. There are 857 proximal arrows in this diagram compared to 10848 of the previous  $\lambda$ -Proxigram.

The diagram clearly illustrates the effectiveness of filtering short proximal arrows, from which we can easily see the source and target points for each tear-up proximal arrow. The diagram shows that most of the tear-up proximal arrows are initiated from the points that lied on the boundaries of the small “patches” in the scatterplot. Furthermore all tear-up proximal arrows are pink, so they are all *joint* tear-ups. Thus the visualisation has no false-outliers, false-neighbours and subtle tear-ups, which means all points with tear-up proximal arrows are surrounded by their true neighbours from HD space.

Fig. 5.5 is the bundled version of the previous Tear-up Proxigram in Fig. 5.4. After applying Mingle algorithm, there are 441 edges compared to 857 of the previous one, which is significantly less cluttered. The tear-ups proximal arrows on the boundaries of the patches are deformed and bundled, revealing the high level patterns that the Gaussian sphere is systematically split into patches by t-SNE.

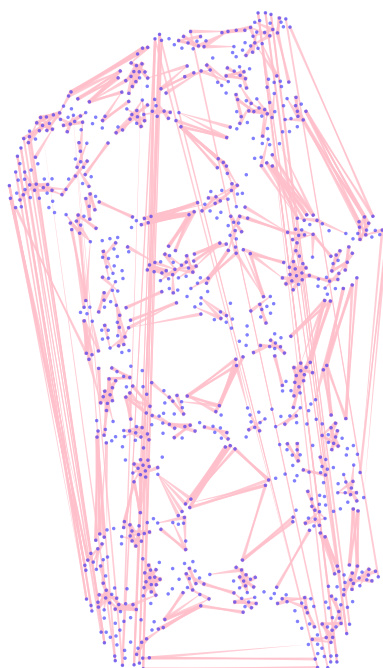


Fig. 5.4 The tearup-Proxigram applied to Fig. 5.3, INES-SN vs DN(2)

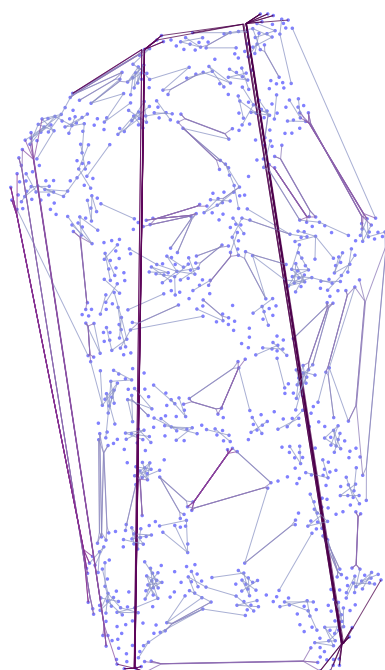


Fig. 5.5 The bundled tearup-Proxigram of the t-SNE visualisation

## 5.5 Spanner Graph from High Dimensional Data

### 5.5.1 Overview

A geometric spanner graph  $G_\eta$  with a sketch factor  $\eta$  is a sub-graph of a weighted graph  $G$  such that the weight of edge or the sum of weights of the connected shortest path of edges of every pair in  $G_\eta$  is at most  $(1 + \eta)$  times the weight of corresponding edge in  $G$ . Such graphs were discovered by [Chew \(1986\)](#).

High dimensional data  $X$  with  $n$  vectors possesses a  $n \cdot n$  latent complete graph  $D$ , its pairwise distance matrix. Our pursuit of constructing an overlay graph is to select a suitable subset of relationships from the latent graph from high dimensional data  $X$  in order to help our eye to recover the structure of high dimensional data in the visualisation. The edges selected by a spanner graph seem to be the “perfect” subset of relationships to approximately represent the complete graph  $D$  from  $X$  and being a sparse graph, from which we expect to recover all pairwise distances and necessary structure. [Chew \(1986\)](#) and subsequent theorists have shown that spanner graphs are surprisingly sparse.

In this section, I will discuss a fraction of details of the well-studied topic, Geometric Spanner Graph and illustrate why it seems to be an amazing overlay graph and why it is also a terrible overlay graph.

### 5.5.2 Greedy Spanner Graph

The definition of geometric spanner is only a description of its core constraint, *sketch factor*, which is also the property of interest for us. There are different kinds of geometric spanners each with different theorems and properties, but constructing the spanner efficiently is not trivial. Therefore we start from a naive approach called *the greedy spanner*.

The greedy spanner algorithm is simple but computationally expensive, and uses algorithm [Dijkstra \(1959\)](#) to repeatedly update all shortest paths. Given a complete graph  $G := \{V, E\}$  with  $V$  being all the points in the space and  $E$  being their pairwise distances  $d_{ij}$ . By initialising the spanner graph  $G_\eta := \{V, \emptyset\}$  and sorting all pairs from the original graph  $G$  by their weight in ascending order  $E'$ , the algorithm begins repeatedly adding an edge from  $E'$  to  $G_\eta$  according to whether the edge satisfies the sketch factor condition or not. If adding the edge  $(a, b)$  makes any pair  $(i, j)$  in the current  $G_\eta$  satisfies  $(\text{dijkstra}(i, a) + d_{ab} + \text{dijkstra}(b, j)) \leq (1 + \eta) \cdot d_{ij}$  that previously it doesn't, then the edge  $(a, b)$  is added. The algorithm terminates till  $G_\eta$  has enough edges to satisfy to the sketch factor condition, [Althöfer et al. \(1993\)](#). In order to make  $G_\eta$  a connected graph in the beginning, our solution is to simply apply minimal spanning tree

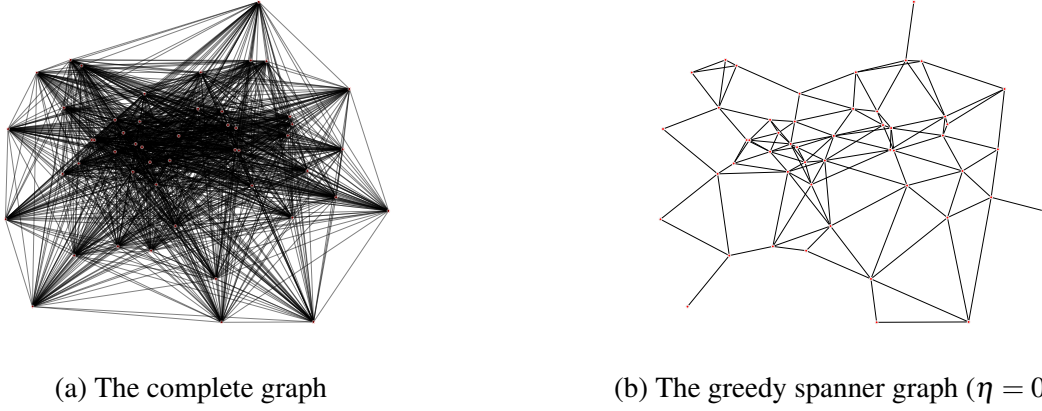


Fig. 5.6 Spanner Example: 50 random points in 2D

[Borůvka \(1926\)](#) to the original graph,  $G_\eta \leftarrow \text{MST}(G)$ , which can help the searching of the algorithm a little bit heuristic.

The lower the sketch factor is, the denser the greedy spanner graph is and longer the algorithm runs. When we tested our implemented greedy spanner algorithm,  $\eta = 0.3$  or 1 is good enough. A sketch factor is too low could result the spanner graph never be found, due the limitation of greedy strategy. We have not explored the idea of the geometric spanner further than the naive greedy spanner (such as efficient algorithm or lower bound), because it turned out that the spanner graph is visually uninterpretable when it is applied to high dimensional data and displayed as an overlay graph.

Here I give the greedy spanner graph applied to two 2D synthetic datasets – a collection of random points in the 2D space and a collection of points forming a circle in the 2D space. Fig. 5.6b is the greedy spanner graph ( $\eta = 0.5$ ) of a collection of 50 points of normal distribution in 2D space and Fig. 5.7b is the greedy spanner graph ( $\eta = 0.3$ ) of a collection of 25 points of a circle in 2D space. They are all relatively sparse compared to the complete graphs from which they are built from (see Fig. 5.6a for 2D random points and Fig. 5.7a for 2D circle points).

Initially we expected to colour all the edges with their HD distance values in the greedy spanner and use it as an overlay graph. It has four difficulties: firstly, a spanner graph, no matter which kind, is not intuitive to read even in the 2D space; Secondly, the situation is worse when displaying the edges of spanner graph of HD data on 2D space as the shortest path between any pair is totally unnatural to look at; Thirdly, we can not visually estimate distance of a pair by adding up the colours of the shortest path; finally, the greedy spanner algorithm is unrealistic to apply to any data with its size larger than 500.



(a) The complete graph

(b) The greedy spanner graph ( $\eta = 0.3$ )

Fig. 5.7 Spanner Example: 25 points Circle in 2D

### 5.5.3 Short Conclusion

The geometric spanner graph seemed to be a promising overlay graph when we thought about it, but it turned out to be very disappointing. However there are a few lessons we have learnt which are proved to be valuable later:

- The edges selected by an overlay graph should take information suggested by the 2D scatterplot layout in some way, which the spanner graph totally ignores. Because the 2D embedding is meaningful as it is produced from dimensionality reduction according to some criteria. An overlay graph that does not use this information would appear redundant at best or even horrifically strange.
- The idea of approximating a complete graph by spanner network and its shortest paths is interesting, but we should consider 2D spanner. But which 2D spanner graph to use and how to use 2D spanner to reflect HD structure? Delaunay plot in Chapter 4 is an elegant and successful spanner in this case.
- Spanner graph or minimum spanning tree significantly reduces the number of edges in complete graph. It could be useful to apply the idea of spanner on a sub-graph of the latent complete graph of HD data. But which sub-graph to apply to? The next section introduces Faithful Proximity Forest, an overlay graph consists of a set of minimum spanning trees in the 2D space.

## 5.6 Exploring Mutual Proximities in Proxigram

The space possesses rich structures if we impose constraints. This section I present a greedy algorithm to find a special kind of structure in the input neighbourhoods, *Mutually Proximate Closure* (MPC), which is similar to *Faithful Delaunay Patch* (see section. 4.8.3), but presented as a completely different overlay graph.

### 5.6.1 Mutual Proximity Closure

Formerly for a set of high dimensional data  $X$  with  $n$  vectors, let  $P := \{P_1, P_2, \dots, P_n\}$  be its input neighbourhood estimation and let  $P_i := \{i, j_1, j_2, \dots, j_s\}$  be the input neighbourhood set of a typical point  $i$ , with its **own** inside. A pair  $(i, j)$  is an asymmetric and reflexive proximity relationship in the set of  $\{(i, j) | j \in P_i\}$ . A pair  $(i, j)$  can only be symmetric and reflexive proximity relationship if the condition  $i \in P_j \wedge j \in P_i$  holds. The two points  $i$  and  $j$  are mutually proximate to each other, forming a closure  $\{i, j\}$  which is a subset of  $P_i \cap P_j$ . Generalising this concept, assuming there is a set of points  $C := \{a_1, a_2, \dots, a_s\}$  that are mutually proximate to each other, the closure must be a subset of  $M := \bigcap_{i \in C} P_i$ .

As discussed in section. 4.8.3, such MPC has many combination possibilities, by which it means the initial configuration and searching orderings will determine the final memberships of the closure. Therefore I propose a greedy strategy to find it.

### 5.6.2 A Greedy Strategy of Searching Mutual Proximity Closures

Intuitively, the first MPC  $C_a$  to find is initialised as  $\{a_1\}$ , the point  $a_1$  with  $|P_{a_1}|$  to be the largest in  $P$  and  $M_a \leftarrow P_{a_1}$ . And then the algorithm gradually builds up  $C_a$  by performing a sequence of set-intersection  $M_a \leftarrow M_a \cap P_{a_i}$  and adding  $P_{a_i}$  to  $C_a$ , until the moment that there are no candidates left which can make  $C_a$  bigger. The candidates to add in  $C_a$  must be the elements in  $M_a$ . However the ordering of adding from the second point would determine the final outcome of  $C_a$ . The algorithm is **greedy** because it always chooses the candidate  $a_i$  that has non-empty and the largest intersection with  $M_a$  **and** it must be a superset of  $C_a$ . Finding the first MPC  $C_a$  must be terminated because there are limited amount of candidates in  $M_a$  and  $M_a$  gets smaller during intersection iteration. Also to note that  $i \in P_i$  holds for every input neighbourhood, the sequence of adding and intersection yields  $C_a = \{a_1, a_2, \dots, a_s\}$  being a subset of  $M_a$ .

After  $C_a$  is found, other MPCs could exist as well. The problem is other closures might overlap with  $C_a$ , which leads to two difficulties: firstly it is a long pursuit since every point can be initialised as closure and the same process applied (which might be interesting in abstract sense) and secondly overlapping closures are hard to draw as overlay graph, because



ultimately the symmetric proximity relationships are drawn as edges. In terms of visualisation, compromise has to be taken for such overwhelming information. The rest of the algorithm implements an even strong constraint, which is the closures to find must be **disjoint**.

In order to achieve this constraint, two measures are taken. Firstly, the process keep a record of the points that are ever added in any previous closures, which are not going to be considered again in building new closures. Secondly, the trace of those points that are ever joined any closures before must be erased (the discrete space  $P$  is altered as  $P'$ ), by which the algorithm will remove those points from all the neighbourhoods of remaining points. The process of finding remaining closure must be terminated because the process of finding each closure must be terminated and there are limited amount of remaining points.

The ordering of finding the remaining closures matters the segmentation of the discrete space. The algorithm is greedy because it always chooses the point that has the largest “altered” input neighbourhood in  $P'$ . Some closures are indeed non-overlapping and the outlier would not have any points but itself within its closure if the input neighbourhood estimation is “appropriate”. However due to the greedy strategy, some points from the “outskirts” of a tight cluster would not have a chance to be within any closure or forming a closure by themselves.

### 5.6.3 Minimum Spanning Forest of Mutual Proximity Closures

After the greedy searching process, a set of disjoint MPCs are found. The remaining problem now is how to draw them as an overlay graph? Usually in normal diagrams, closures are drawn as convex hulls or in distinguishing colours to indicate the memberships of points. The first option is not feasible in high dimensional visualisation because the points close to each other in the closure would not necessarily remain close in the scatterplot, drawing convex hulls will no doubt overlapping and covering each other. The same reason applied to the second option and also there are not enough distinguishable colours for possibly many closures. Proximity relationship in this chapter are presented as proxi-arrows. A MPC with  $t$  points has  $t \cdot (t - 1)$  proximate proxi-arrows or  $t \cdot (t - 1)/2$  edges since they are symmetric proximity relationships. However there is a better solution to this. High dimensional data  $X$  and its embedding  $Y$  define latent complete graphs (undirected) like the pairwise HD Euclidean distance matrix  $D$  and 2D Euclidean distance matrix  $\hat{D}$ . A closure with  $t$  points inside defines a sub undirected complete graph ( $t \cdot t$  distance matrix), from which Minimum Spanning Tree (MST) can be drawn. Because the  $t \cdot (t - 1)/2$  edges within the closure just mean one thing: the  $t$  points are mutually proximate. MST drawn from the sub-complete graph of the closure connects all the points which is enough for the eye to distinguish the closure from others. Also the closures are disjoint, thus all the minimum spanning trees are disjoint, which forms a minimum spanning forest. But which distance matrix to be used  $D$  or  $\hat{D}$ , since they are both applicable in principle.

We prefer Proxigram to have short proxi-arrows as many as possible because it is less clutter. The sum of edges' weights (Euclidean distance values in this case) in MST is minimum. Thus MST connects all the points in the MPC using the shortest edges as many as possible. In order to reduce the clutters in overlay graph, the 2D pairwise Euclidean distance matrix is favourable over the HD distance matrix, because using  $\hat{D}$  let MST select the shortest edges in 2D space as many as possible. After obtaining the minimum spanning forest of mutual proximity closures, a random RGB colour is given for each tree. The overlay graph produced from algorithm GMP-MSF is called *Faithful Proximity Forest* (FPF), which is a subset of Proxigram.

Finally the greedy algorithm called *Greedy Mutual Proxi Minimum Spanning Forest* (GMP-MSF) to find disjoint MPCs in  $P$  and building minimum spanning forest is presented in Algorithm. 6. Notations in the algorithm are consistent with the previous discussion. The only input parameters for the GMP-MSF is  $\hat{D}$  the 2D Euclidean distance matrix and  $P$  the input neighbourhood estimation of  $X$ . The output of the algorithm is  $F$  the minimum spanning forest of Cs the MPCs founded. `checked` is a boolean array that keeps record of points that ever joined a closure in previous search. `updateP(P,C)` is the function responsible for erasing all trace of all the points in the new founded closure from  $P$ . It performs  $P_i \leftarrow \emptyset, \forall i \in C$  and  $P_i \leftarrow P_i \setminus C, \forall i \notin C$ . `buildSubGraph( $\hat{D}, C$ )` create a sub distance matrix of  $\hat{D}$  based on given points in  $C$ .

#### 5.6.4 Properties of Faithful Proximity Forest

The Faithful Proximity Forest is a segmentation of the discrete space of input neighbourhood with information from the output space, while the Faithful Delaunay Patches map is a segmentation of the discrete space of the output neighbourhood with information from the input space. Both overlay graphs present similar discrete space structures, the mutual proximity relationships. The latter one focuses on discovering the transitive property of Delaunay edges on the 2D scatterplot, so the triangles are connected, while the Faithful Proximity Forest does not care whether the mutual proximity points found are actually visualised close to each other or not. Of course if the mutual proximity closure were visualised well, its points should be placed closely in the scatterplot, so that the Delaunay edges that connect them would exhibit transitive properties. In this sense, Faithful Proximity Forest displays the mutual proximity closure that the input neighbourhoods **should** possess and Faithful Delaunay Patches Map shows the mutual proximity closure that the output neighbourhoods **could** exhibit.

Faithful Proximity Forest is a subset of General Proxigram and Faithful Delaunay Patches Map is a subset of Delaunay Plot. Both overlay graphs holds strong constraints with a few points left without any annotation. The outskirts points and outliers are indistinguishable in Faithful Proximity Forest; While if two nearby points are close to each other in the HD space

**Algorithm 6** Greedy Mutual Proxi Minimum Spanning Forest**Input:**  $\hat{D}, P$ **Require:**  $n \leftarrow$  the number of total points.**Require:**  $Cs \leftarrow \{\}$ **Require:**  $F \leftarrow \{\}$ **Require:**  $checked \leftarrow \{\text{false}, \dots, \text{false}\}$  in the size of  $n$  $s \leftarrow \operatorname{argmax}_{i=1}^n \{|P_i| : P_i \in P\}$  $checked[s] \leftarrow \text{true}$  $M \leftarrow P_s$ **while true do****while true do** $C \leftarrow \{i | i \in M, checked[i] = \text{true}\}$  $candidates \leftarrow \{i | i \in M, checked[i] = \text{false} \wedge C \subseteq P_i\}$ **if**  $|candidates| = 0$  **then****break****end if** $relativeIdx \leftarrow \operatorname{argmax}\{|M \cap P_i| : i \in candidates\}$  $bestIdx \leftarrow candidates[relativeIdx]$  $M \leftarrow M \cup P_{bestIdx}$  $checked[bestIdx] \leftarrow \text{true}$ **end while****if**  $|C| > 1$  **then**add  $C$  to  $Cs$ **end if****if**  $checked$  is not all **true** **then** $P \leftarrow \text{updateP}(P, C)$  $idxs \leftarrow$  a set of all positions in  $checked$  that are **false** $relativeIdx \leftarrow \operatorname{argmax}\{|P_i| : i \in idxs\}$  $s \leftarrow idxs[relativeIdx]$  $checked[s] \leftarrow \text{true}$  $C \leftarrow P_s$ **else****break****end if****end while****for**  $C \in Cs$  **do** $\hat{D}' \leftarrow \text{buildSubGraph}(\hat{D}, C)$  $T \leftarrow \text{minimumSpanningTree}(\hat{D}')$ add  $F$  to  $T$ **end for****return**  $F$

but not for the three nearby points, there is no patch is drawn in Faithful Delaunay Patches Map.

Due to the strong constraints FPF holds, there are some special properties:

- The points connected by Minimum Spanning Tree are mutually proximate to each other. So they are in each other's input neighbourhoods. Such structure is very likely to be a tight cluster or a patch of a manifold in the high dimensional space.
- A short edge in one tree means the two points are visualised closely in the scatterplot. The more short edges the tree has, the better the closure is visualised.
- A tree that has only short edges means the closure is visualised faithfully.
- A long edge in one tree is the **critical** tear-up, it can be the one of the four kinds of tear-ups defined in Eq. 5.10.
- The Faithful Proximity Forest significantly reduces the visual clutter, as it uses the smallest amount of “ink” to draw the overlay graph.

Here I present an example of faithful proximity forest (FPF). Fig. 5.8 is the FPF of the t-SNE visualisation ( $perplexity = 30$ ) of the 3D Gaussian sphere dataset with the input neighbourhood estimation using kNN ( $k = 30$ ). Each minimum spanning tree indicates a mutual proximity closure found in the dataset. The long spanning tree edges in the diagram are the proofs that the FPF is indeed capable of revealing the tear-up distortions that within the mutual proximity closures and each such tear-up spanning tree edge is shared by the closure members. Surprisingly the diagram is similar to the bundled tearup-Proxigram in Fig. 5.5 in a more meaningful way, as the mutual proximity closure holds a strong constraint. However due to the constraint, some tear-up distortions in tearup-Proxigram are excluded. And because of the greedy searching strategy, some points are left alone. The faithful proximity forest helps user to find the high “level of truth” tear-ups automatically and these tear-up spanning tree edges must be the *joint* type tear-ups.

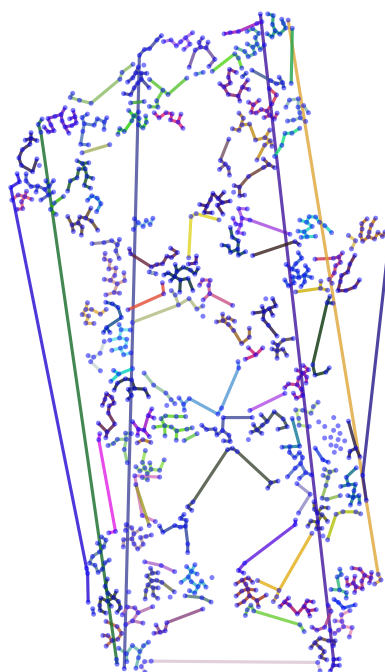


Fig. 5.8 The Faithful Proximity Forest of the Gaussian Sphere, INES-kNN( $k = 30$ )

## 5.7 Conclusion

The General Proxigram is the overlay graph that holds our whole assumption of input neighbourhood estimation, even imperfect. It is a simple but rather primitive and cluttered visual representation of proximity relationships because it does not take output space neighbourhood information into consideration. Throughout the chapter we aim to provide a subset of General Proxigram in order to reduce the number of unnecessary proxi-arrows and improve the readability, thus  $\lambda$ -Proxigram, Tearup-Proxigram, and Faithful Proximity Forest are introduced, with reduction the number of proxi-arrows and increased readability. Each kind of Proxigram has its own purpose, from which the true structure of high dimensional data is gradually presented. Proxigrams of any kind can not directly indicate false-neighbours, though those points responsible for triggering false-neighbours are likely to initiate tear-ups.

No overlay graph can present all proximity relationships of interest and all distortions we care about. The next chapter introduces the final overlay graph, *Tapestry Plot* to alleviate this problem.

# Chapter 6

## Tapestry Plot

Revealing more information about high dimensional data requires displaying more relationships. The efficiency of visualisation is difficult to measure because we have exquisite eyes. Sometimes showing more information does not have to be a trade-off of readability if the visualisation is carefully designed.

The Delaunay Plot is a diagram of *blurred* meshes underneath the scatterplot, radiating ambient messages of symmetric proximity relationships among the nearby points and revealing the worst false-neighbour distortions; The Proxigram is a diagram of *sharp* arrows on top of the scatterplot, emitting irrefutable signals of the asymmetric proximity relationships of all points in the HD space and revealing the tear-up distortions. The two overlay graphs are in fact complementary to each other. This chapter introduces a more expressive overlay graph termed as *Tapestry Plot*, consisting of a combined Tear-up Proxigram and Delaunay Plot. We aim to use this overlay graph to reveal both tear-up and false-neighbour, while the proximity signals from short proxi-arrows are replaced by Delaunay edges and their transitive properties.

With Delaunay Plot and Tearup-Proxigram fully established in previous chapters, the introduction of Tapestry Plot is simple. The remainder of this chapter presents how the traditional visualisation quality measures are changed due to shown tear-up and false-neighbour, the control of uncertainty in the Tapestry plot, a recommended work-flow for visualising high dimensional data in general and several high dimensional data visualisation sequences as showcases, from which the interpretation of the HD data becomes clear.

### 6.1 Control of Uncertainty

The scatterplot visualisation of high dimensional data has many distortions even it excels in visualisation quality measure like Precision & Recall [Venna et al. \(2010\)](#), which involves considerable level of uncertainty. Tapestry Plot shows both tear-up and false-neighbour distortions.

tions. Since these distortions are shown (corrected), they are no longer errors in visualisation, hence the uncertainty level decreases. Thus we derive a visualisation quality measure based on Precision & Recall in respect of Tapestry Plot to indicate how many distortions are left uncorrected, since high order tear-ups and false-neighbours are still invisible.

Tapestry Plot requires  $\lambda$ -Proxigram and Delaunay Plot. Given high dimensional data  $X$  with  $n$  items, an input neighbourhood estimation  $P = \{P_1, P_2, \dots, P_i, \dots, P_n\}$  from any estimation scheme and an output neighbourhood defined by Delaunay Neighbour  $Q^\phi = \{Q_1^\phi, Q_2^\phi, \dots, Q_i^\phi, \dots, Q_n^\phi\}$  with  $\phi$  being a positive integer, the length of path user wants to observe as introduced in section. 4.6.1. Also let  $P^\lambda = \{P_1^\lambda, P_2^\lambda, \dots, P_i^\lambda, \dots, P_n^\lambda\}$  be the input neighbourhoods in respect of level of truth as introduced in Eq. 5.5 with  $\lambda$  being a real number ranging from 0 to 1. Formally the definition of the modified Precision & Recall is given in Eq. 6.1.

$$\text{Precision}_i := 1 - \frac{|Q_i^\phi \setminus P_i| - |Q_i^1 \setminus P_i|}{|Q_i^\phi|} = 1 - \frac{\text{false-neighbour not shown}}{\text{size of output neighbourhood}} \quad (6.1)$$

$$\text{Recall}_i := 1 - \frac{|P_i \setminus P_i^\lambda \setminus Q_i^\phi|}{|P_i|} = 1 - \frac{\text{tear-up not shown}}{\text{size of input neighbourhood}} \quad (6.2)$$

Since the worst false-neighbour are revealed by Delaunay edges, Precision will no doubt be improved; While low rank (within level of truth) tear-ups are revealed by proxi-arrows, Recall will be improved considerably. However I argue that visualisation quality measures are only useful to index the performance of DR methods. The most valuable theoretical instruments from Information Perspective of HD visualisation [Venna et al. \(2010\)](#) are the general definitions of input and output space neighbourhood which helps to identify tear-ups and false-neighbours that overlay graphs can select. Tapestry Plot makes the traditional quality measures trivial, because in Tapestry Plot, most of the distortions are already shown and if the distortions are too many, the visualisation is not trustworthy. In this sense our focus should shift from quality measures to the real task of visualisation: how to successfully use Tapestry Plot to retrieve similarity information from high dimensional data, which depends on how much uncertainty introduced.

There are a variety of factors for the uncertainty in Tapestry Plot. To dissect them, we need to look at what are the points “related” to a typical point  $i$  in general sense. Given an input neighbourhood estimation  $P$ , its level of truth  $\lambda$  and output neighbourhood  $Q^\phi$  defined by Delaunay neighbours with  $\phi$  being the length of the shortest path. The whole set of points that related to point  $i$  is  $W_i = P_i \cup Q_i^\phi$ . The dissection is written as below:



$$P_i \cup Q_i^\phi = (P_i \setminus Q_i^\phi) \cup Q_i^1 \cup (Q_i^\lambda \setminus Q_i^1) \quad (6.3)$$

$$= \left( P_i^\lambda \cup (P_i \setminus P_i^\lambda) \right) \setminus Q_i^\phi \cup Q_i^1 \cup (Q_i^\lambda \setminus Q_i^1) \quad (6.4)$$

$$= \left( P_i^\lambda \setminus Q_i^\phi \right) \cup \left( (P_i \setminus P_i^\lambda) \setminus Q_i^\phi \right) \cup Q_i^1 \cup (Q_i^\lambda \setminus Q_i^1) \quad (6.5)$$

in which  $P_i^\lambda \setminus Q_i^\phi$  is the set of tear-up proxi-arrows shown (certainty);  $P_i \setminus P_i^\lambda \setminus Q_i^\phi$  is the set of tear-up proxi-arrows not shown (uncertainty);  $Q_i^1$  is the set of any relationship that are revealed by Delaunay edges (certainty); And finally  $Q_i^\lambda \setminus Q_i^1$  is the set of points that are not directly connected with point  $i$  with mixed of truth neighbours and false neighbours.

The uncertainty of the tear-up proxi-arrows not shown is not interesting, because we choose not to display them. However the uncertainty in  $Q_i^\lambda \setminus Q_i^1$  is a mystery, because there are no indications of true neighbours and false neighbours (either because short proxi-arrows are filtered or hidden false neighbours etc.) when the length of the shortest path above 1, which is high-order relationships that Tapestry Plot can not show.

Imagine a point  $i$  in the scatterplot and the Delaunay mesh surround it (size is defined by the shortest path length  $\phi$ ), the nearby points are mixed with high-order true neighbours ( $j \in (Q_i^\phi \setminus Q_i^1) \cap P_i$ ) and high-order false neighbours ( $j \in (Q_i^\phi \setminus Q_i^1) \setminus P_i$ ), which is a complete set division of  $Q_i^\phi \setminus Q_i^1$ .

The successfulness of retrieving high-order similarity relationships is high when high-order true neighbours are the majority in  $Q_i^\phi \setminus Q_i^1$  (the surround mesh) and if high-order true neighbours are fully occupies the mesh, the uncertainty of high-order false neighbours is also none; While If high-order false neighbours are the majority of the mesh, the chance of predicting high-order false neighbours is high but for low for high-order true neighbours, but if high-order false neighbours fully occupy the mesh, the uncertainty of high-order true neighbours is also none. Therefore the situation can be described as a pair of indicators: the number of high-order true neighbours in the mesh surround the centre point and the number of high order false neighbours in the mesh surround the centre point, the number of both cases adding up to  $|Q_i^\phi \setminus Q_i^1|$  always.

Therefore it is natural to define the overall uncertainty and specific uncertainty of point  $i$  using Information Entropy:

$$H(\phi)_i := -s_{i,1} \log(s_{i,1}) - s_{i,2} \log(s_{i,2}) \quad (6.6)$$

$$H(\phi) := \frac{\sum_{i=1}^n H(\phi)_i}{n} \quad (6.7)$$

$$\text{where } s_i := \left\{ \frac{|(Q_i^\phi \setminus Q_i^1) \cap P_i|}{|Q_i^\phi \setminus Q_i^1|}, \frac{|(Q_i^\phi \setminus Q_i^1) \setminus P_i|}{|Q_i^\phi \setminus Q_i^1|} \right\} = 1 \quad (6.8)$$

The smaller  $H(\phi)$  is, the lower the uncertainty is in Tapestry Plot. This is the measure only for uncertainty, **not** for visualisation quality. With definition of  $H(\phi)$ , we can control the uncertainty involved in Tapestry Plot, from which we can know which part is very mixed with true neighbours and false neighbours. The interactive application with this feature gives user an option of not filtering short proxi-arrows in high uncertainty part such that the uncertainty is diminished.

Uncertainty is a trade-off to readability for Tapestry Plot, because less proxi-arrows are the less clutter the graph is. Our successfulness of retrieving similarity relationships of high dimensional data also depends on readability. Unfortunately readability seems to be a subjective matter, which is hard to measure objectively. Our visual perceptual capacity related to Tapestry Plot is very difficult to describe let alone to define it in mathematics. This is a potentially interesting problem which requires some psychological studies in the future.

It is worth to mention that every faithful Delaunay patch in Faithful Delaunay Patches Map and every minimum spanning tree in Faithful Minimum Spanning Forest does not involve any uncertainty, because all points connected to it are all mutually proximate to each other, hence no false neighbour but all true neighbours. In this sense, Faithful Delaunay Patches Map and Faithful Minimum Spanning Forest are the visual representations of the parts that observer can always trust in the visualisation.

## 6.2 Work-flow of Visualising High Dimensional Data

The overlay graphs introduced in this thesis are only the augmentations to the scatterplot visualisation, revealing what the scatterplot is distorted. Therefore the **core** of visualising high dimensional data is still dimensionality reduction method. Overlay graph of any kind is not worth to observe if the scatterplot visualisation quality is very low, which can be indexed from variety of objective visualisation quality measures. The primary method is Venna's Precision and Recall, with its input neighbourhoods defined by Stochastic Neighbours (see Algorithm. 2) and its output neighbourhoods defined by Delaunay neighbours (see Algorithm. 3). But

another intuitive objective quality measure is Delaunay Plot Cleanness (see Eq. 4.1), because a visualisation is definitely poor in Venna's Precision and Recall criteria if the cleanness is low, meaning the visualisation has too many the worst false-neighbours.

PCA is perhaps the first DR method to try, for its simplicity and clear intuition, from which some structures in respect to variation are exhibited by the two principal components. But very often the classical linear projection would fall in terms of visualisation quality measures. MDS family methods are only suitable for dataset with its dimensionality very low but above 2 of course, because the mismatch of HD distance and 2D distance becomes meaningless as the dimensionality increasing. Manifold learning methods like LLE and Isomap are suitable for dataset with known low dimensional manifolds, they are not robust to visualise complex high dimensional data. Finally **t-SNE** is good and robust for the most cases if its perplexity (usually between 30 to 50) is appropriate and the size of data is big enough (more than 500). Thus it is our preferred DR method for producing the 2D embedding currently.

Unless we have prior knowledge of the dataset distribution and it is regular, k-Nearest Neighbours and HD Euclidean distance radius are not suitable as Input Neighbourhood Estimation Scheme. The estimation scheme Stochastic Neighbours is recommended but to note that its perplexity does not have to be the same as the perplexity of t-SNE (see section. 4.5.4 for parameter guideline). Given an input neighbourhood estimation and a scatterplot visualisation, we can begin by drawing Delaunay Plot.

Delaunay Plot and its Cleanness measure are computational cheap. A DR method that produces the scatterplot with low cleanness can be discard or adjusted quickly just by the Cleanness. When a scatterplot visualisation is verified as reasonably good quality, other overlay graph can be drawn upon. However if all DR methods are tried and all the possible parameters are tried and still Delaunay Plot Cleanness can not be improved (the process which can be automatic just by using Cleanness as the primary optimisation goal), it suggests the dataset is too complex to visualise.

The first overlay graph to draw is General Proxigram, but be aware that even a small number of points are visualised incorrectly, the proxi-arrows could dominate the graph. Thus in order to see the more important proxi-arrows,  $\lambda$ -Proxigram and Faithful Proximity Forest can be applied. These two overlay graph will help us to learn in what level the HD similarities is worth to observe. Faithful Proximity Forest segments the input neighbourhood estimation into mutually proximity closures, from which the size of tight clusters can even be visually identified.

After a favourable level of truth is chosen, we can proceed to Tearup-Proxigram and Tapestry Plot. When tear-up proxi-arrows dominate the visualisation, edge bundling can be applied. Edge bundling shows potential patterns exhibited in the long edges. If some edges are bundled

greatly it means the points responsible for tear-up proxi-arrows are distorted systematically, otherwise not. Finally observers can use an interactive application to explore Tapestry Plot with a variety of sliders to adjust the parameters, from which the similarities in HD data can be gradually learned.

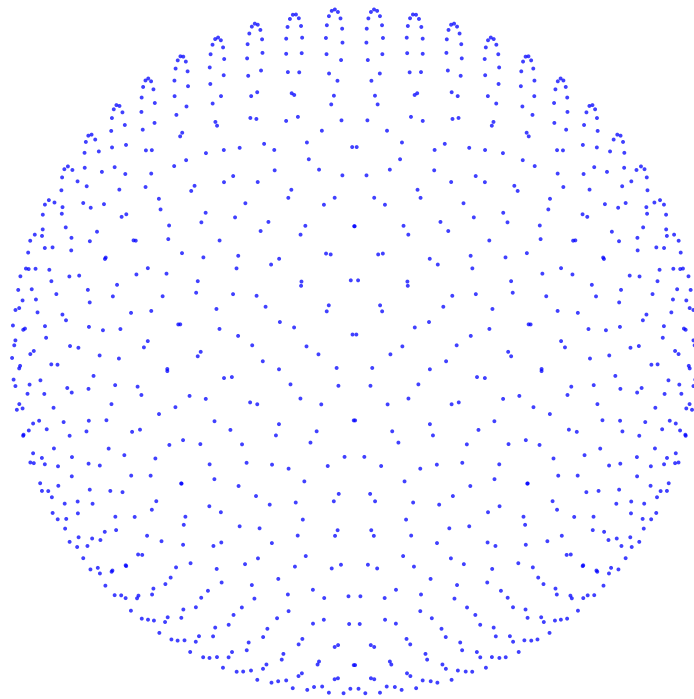
### 6.3 Visualisation Sequences of Spherical Shell

In this section I will use varieties of visualisation techniques to systematically discover the underlying structure of the showcase 3D Spherical Shell dataset presented in Fig. 2.1. A sequence of visualisation will be presented as following:

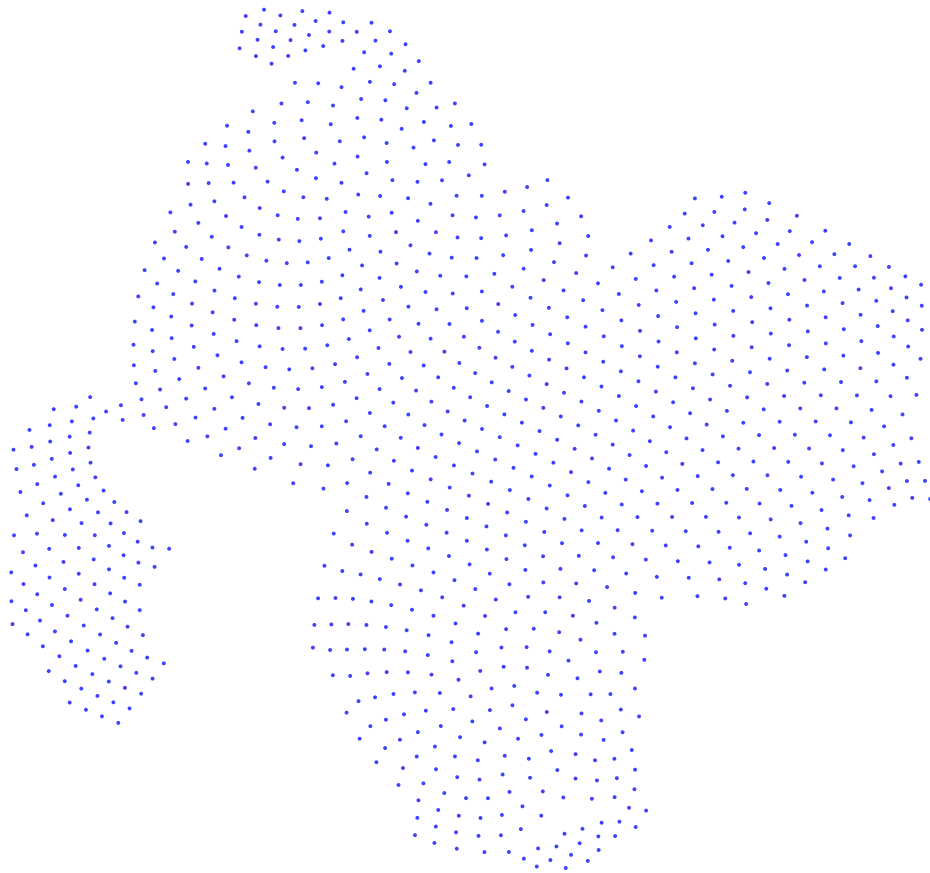
1. Plain scatterplots of DR methods;
2. DR method specific comfort plots;
3. Delaunay Plots;
4. Proxigrams in k-nearest neighbours;
5. Proxigrams in tear-up filtering of above kind;
6. Tapestry Plots;
7. Tapestry Plots with proxi-arrows bundled.

The intention of the visualisation sequence specially in this order has three reasons: firstly it is a self-telling story about how the structure is unfolded and revealed by the mean of overlay graphs only; Secondly it is a suggested work-flow when users have very little knowledge about the data and want to explore structure by visualisation; Thirdly with the sequence proceeding it is clear that the plain scatterplot visualisation, its optimisation error allocated and the visualisation quality measures are far from enough, which are the traditional practice of visualising and examining high dimensional data.

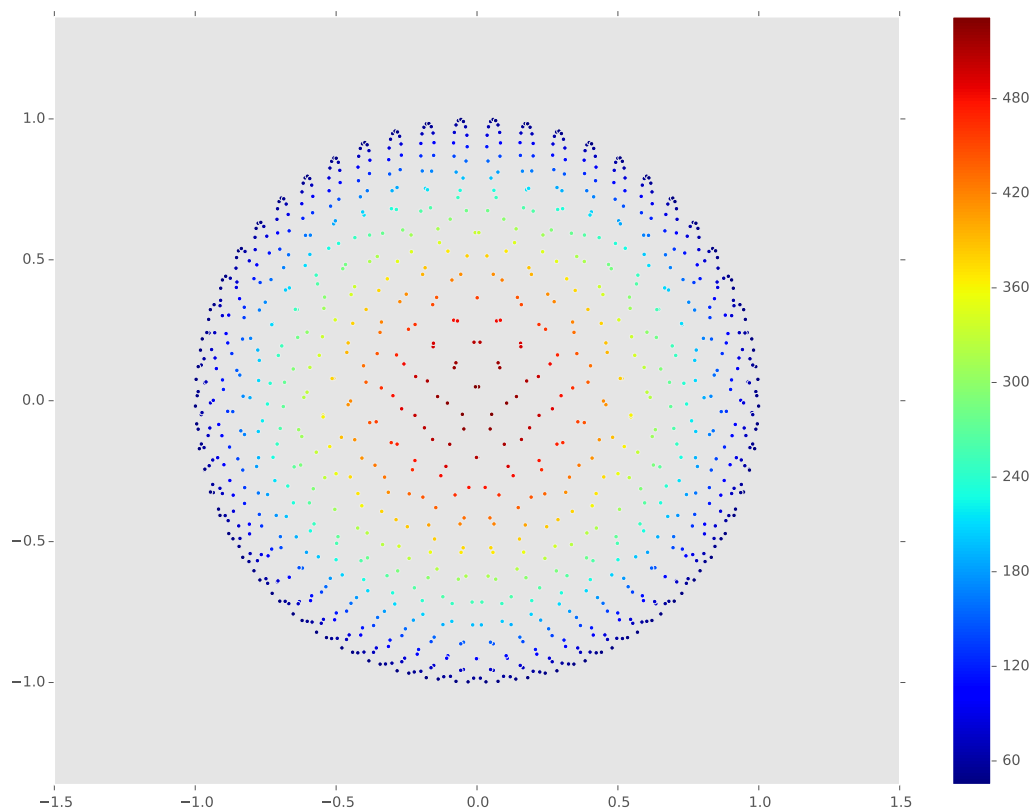
Now let us pretend that all we know about this dataset is that it is a collection of 1000 points in 3D space and begin the journey of visual exploration.



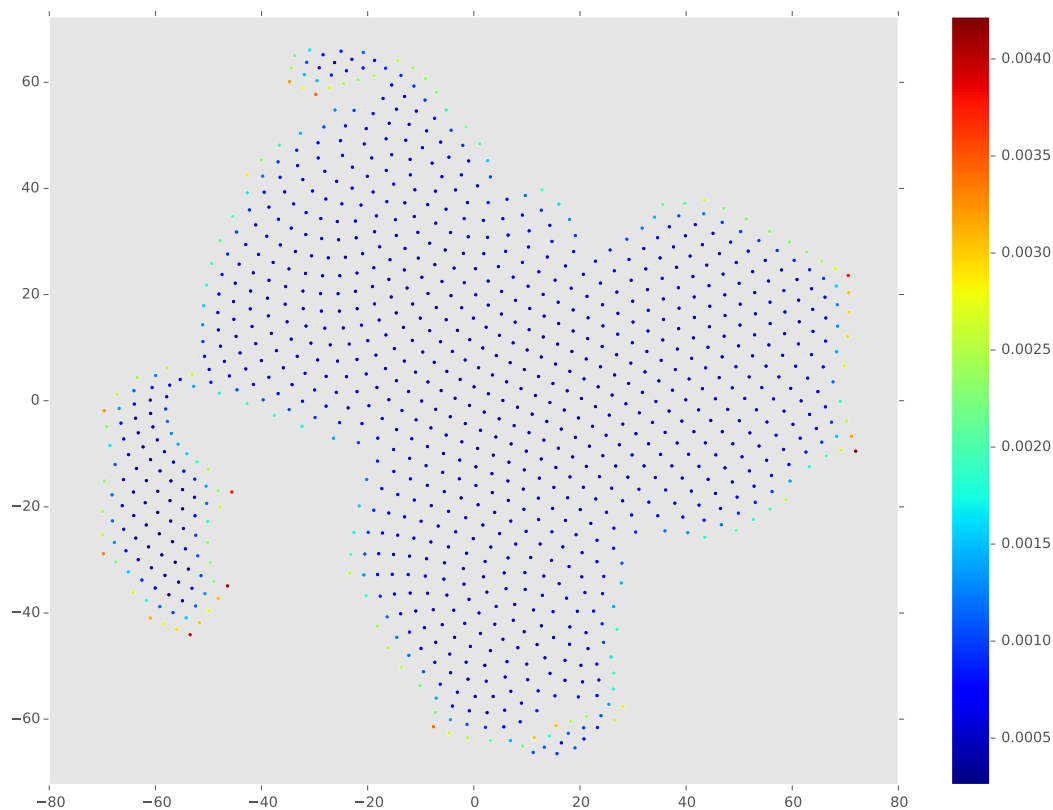
Plain Scatterplot produced by PCA. This is the textbook procedure from dimensionality reduction and visualisation, because it has a very clear geometric intuition. The visualisation actually tell us something that the data in 3D space possibly form a circle or something and quite uniformly distributed, because if there is an outlier, its variance will make it stand out in the scatterplot. But how the points related, are any two points close to each other in 3D space? There is no cue of this.



Plain scatterplot produced by t-SNE (*perplexity* = 30). The X, Y axis in t-SNE visualisation are not interpretable unlike PCA, but the affinities values of  $P$  distribution of t-SNE are built from the 3D pairwise distances of data points. All points in the scatterplot are very uniformly laid indicates that the data is likely very uniformly distributed due to their very identical affinity distribution. There are small patches stretched out, perhaps those points have particular distinct structure than the main patch. But we can neither conclude any part are locally true close in 3D space nor we are confident about the stretched out patches stand out in the 3D space. The scatterplot is very different from the one by PCA. In fact all scatterplots produced by other DR methods (MDS, NLM, LLE and Isomap) are almost identical to PCA. We are not sure t-SNE visualise the data significantly wrongly.

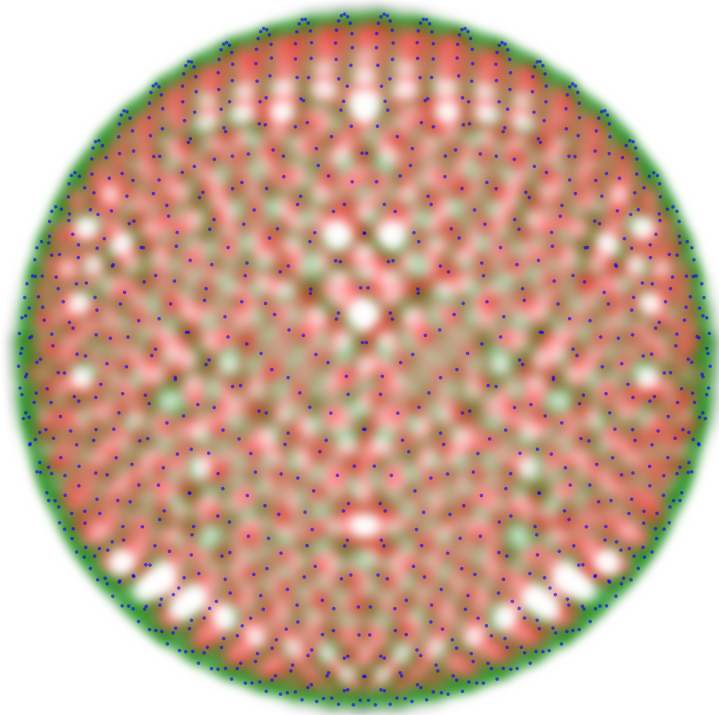


Comfort Plot of PCA scatterplot in MDS Stress. In theory PCA yields the exactly the same result as classical MDS does. Thus PCA can be evaluated by MDS stress, which is the square errors between 3D and 2D pairwise distance. The plot gives us a “circle-symmetric” gradient about the stress. The outer part of the circle has less stress than inner part, which suggest the pairwise distance are not scaled greatly from their original. The inner part however is seriously “uncomfortable”. We know the data has 1000 points, that is to say, the point has stress of 500 is distorted by  $500/1000 = 0.5$  in distance variance, which is half of the scale of the largest pairwise distance in 3D. There are so many uncomfortable points inside the circle and yet there is no cues of where to put them “comfortably”.



Comfort Plot of t-SNE ( $perplexity = 30$ ) scatterplot by colouring the points in the allocated cost function value. The cost function of t-SNE is the cross entropy from distribution  $P$  to  $Q$ , the mismatch between the two distribution contribute the total error. The plot shows almost every points on the boundary of the patches has large contribution to the cost function error, by which it means these points are the most “uncomfortable” ones. But we know there is large positive cost for tear-ups but only small positive cost or possibly negative for false-neighbours. The points inside the boundary stay “comfortable”, which suggests those points are visualised locally well in t-SNE standard. It start to get interesting because those points on the boundary should be the points responsible for tear-ups but how are they distorted?

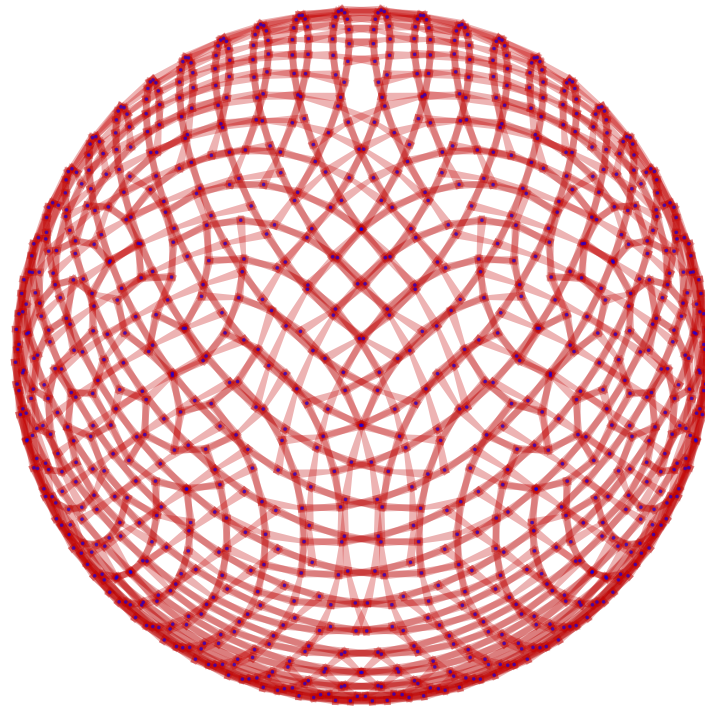




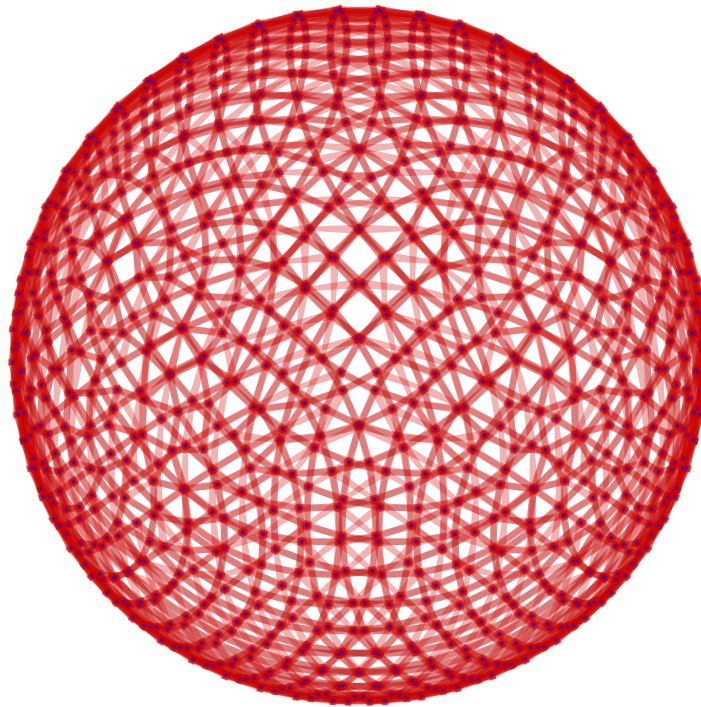
Delaunay Plot of PCA scatterplot (LDECS-SN *perplexity* = 20; INES-SN *perplexity* = 30). The Delaunay edges on the boundary of the circle are all green, indicate that those nearby points are truly close to each other, which Comfort Plot suggests the similar but with weak evidences. However the red and green Delaunay edges inside the circle are interwoven, indicating there are true neighbours of some pairs but the remaining are faraway estimated by INES-SN. This is also confirmed from the previous Comfort Plot. Now we know the scatterplot from PCA has a great many false-neighbour distortions inside the circle.



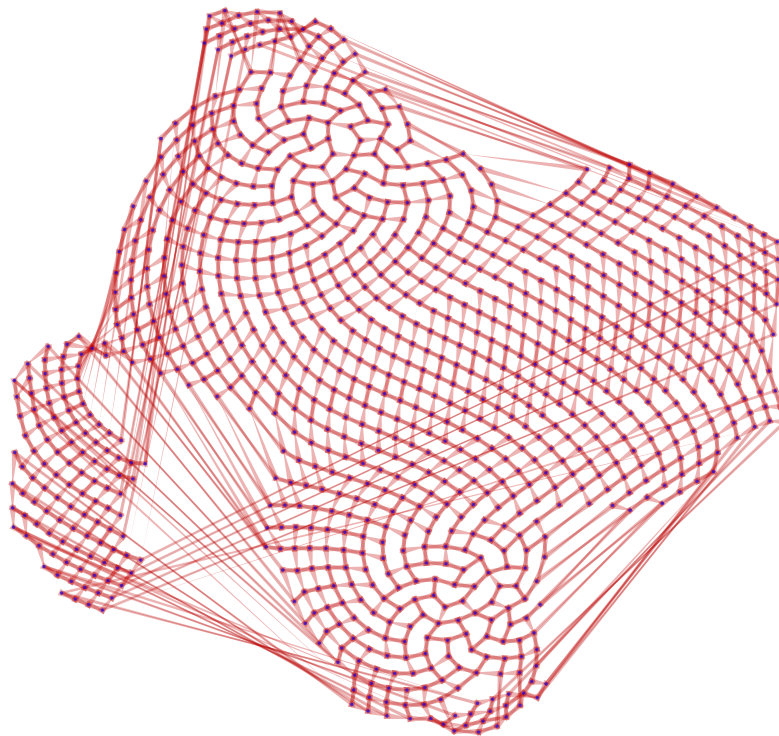
Delaunay Plot of t-SNE scatterplot (input neighbourhood estimation scheme as stochastic neighbours  $perplexity = 30, w = 3$ ). Previously from Comfort Plot of t-SNE we suspect points inside patches might be visualised comfortably, the evidence of which is compelling in Delaunay Plot, as all the edges inside the patch are green except one. Each point on the boundary has Delaunay edges connecting to the points inside the patch, but those are also green. Delaunay Plot shows the t-SNE visualised the local structure of the dataset very well. Delaunay Plot can not show any tear-up errors, which Proxigram aims to address.



General Proxigram (kNN,  $k = 3$ ) of PCA scatterplot. There are three proxi-arrows drawn from each point on the scatterplot and every proxi-arrow is red, which means all input neighbourhoods defined by distance rank 3 are small in 3D distance. The topological order is highly correlated with distance scale, thus k-nearest neighbours scheme is legitimate neighbourhood estimation. The proxi-arrows are all short, meaning there are no tear-ups. The point suffers from false-neighbour distortion usually comes with tear-ups as well, because the 2D space for true neighbours to place is filled with false-neighbours. The proxi-rank of this diagram simply can not go higher because the source-point and end-point on the outer parts of the circle are already not recognisable.

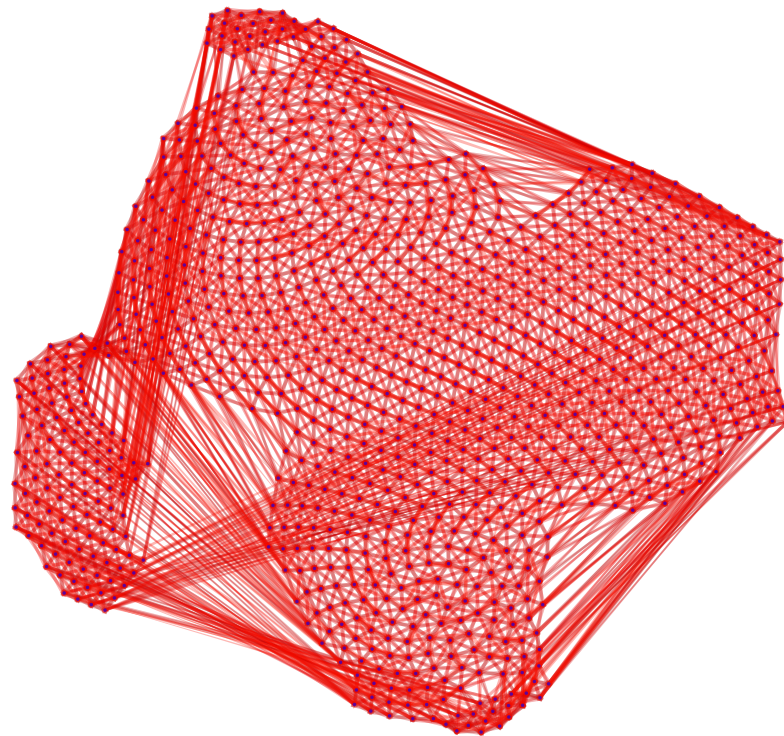


General Proxigram (kNN,  $k = 5$ ) of PCA scatterplot. The proxigram is similar to rank-3 one, there are all short red proxi-arrows, meaning the true neighbours for each points are not far from it in distance scale. Given previous diagrams, perhaps we could know what the underlying structure is, even from this poor readability of this diagram.

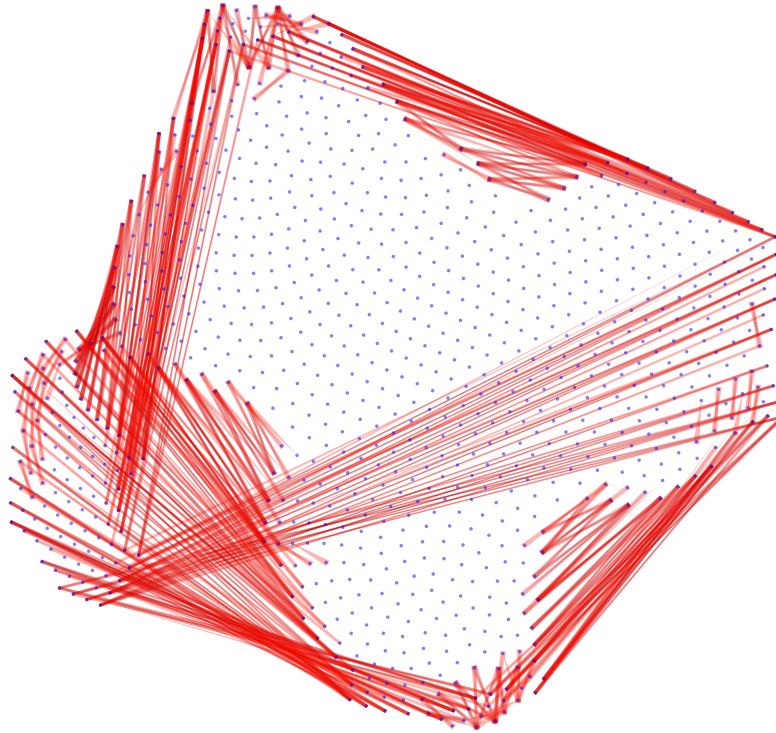


General Proxigram (kNN,  $k = 3$ ) of t-SNE (*perplexity* = 30) scatterplot. The colours of all proxi-arrows are the same as PCA's proxigram because they are same relationships selected. However the layout of this proxigram is much different. There are short red proxi-arrows within the patches indicating that there are no tear-up errors within 3-nearest neighbours for those point. It also gives clear reason why there are mainly green edges in Delaunay Plot. But nearly all points on the boundary of the patches have at least one long red proxi-arrow meaning that some of their true neighbours are on the other side of the boundary. The evidence is also suggested by t-SNE Comfort Plot. Now we know which points are close to which within 3-nearest neighbours. But in order to examine larger input neighbourhood and more possible tear-ups, the proxi-rank must be higher.

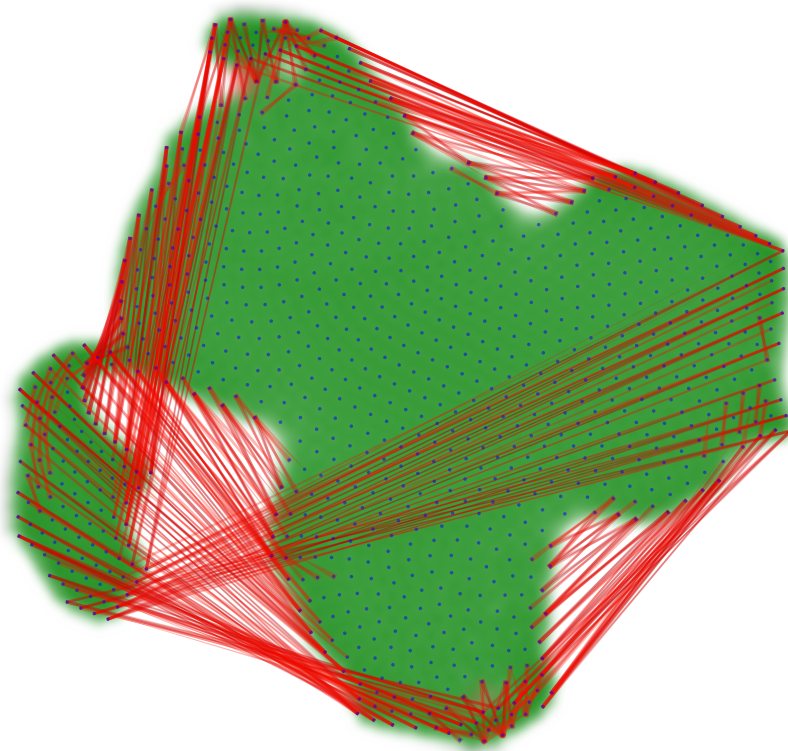




General Proxigram (kNN,  $k = 10$ ) of t-SNE (*perplexity* = 30) scatterplot. t-SNE gives a promising layout that have nice Delaunay Plot and Rank-3 Proxigram. Rank-10 Proxigram means there is 10 arrows drawn from every point. This diagram has similar pattern as Rank-3 Proxigram, the arrows inside the patches forming a grid like structure and the points on the boundary have many reaching out arrows pointing to another side of the boundary. The readability is not that poor for a rank-10 Proxigram, which means the scatterplot has systematic distortions; In order to improve the readability further, Tear-up Proxigram must be drawn next.

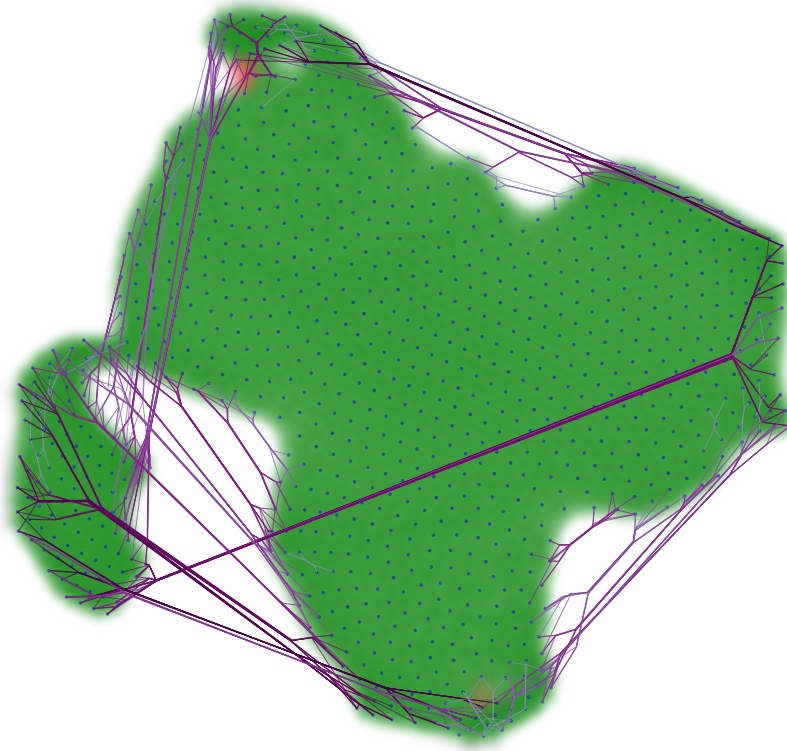


Tearup-Proxigram (kNN,  $k = 10$ ) filtering by Delaunay neighbour reachable  $\phi = 2$  of t-SNE (*perplexity* = 30) scatterplot. Proxi-rank higher than 3 is sometimes useful to explore and recover the underlying structures. But proxi-rank 10 is too many to display in Fig. 6.3. The tear-up filtering scheme will trim out the proxi-arrow if its source point and target point are reachable by travel only two connecting green Delaunay edge (Delaunay Rank 2). The diagram has almost all proxi-arrows trimmed meaning, meaning every point in the patches are truly close with other points in Delaunay rank 2 range. Thus there is no need for those proxi-arrows. However boundary points still have many long proxi-arrows, meaning their true neighbours are out of reach of the output space neighbourhood defined by Delaunay neighbour reachable. Lastly the Delaunay neighbour reachable is not visible in this diagram due to lack of the Delaunay plot. The tapestry plot should be drawn next.

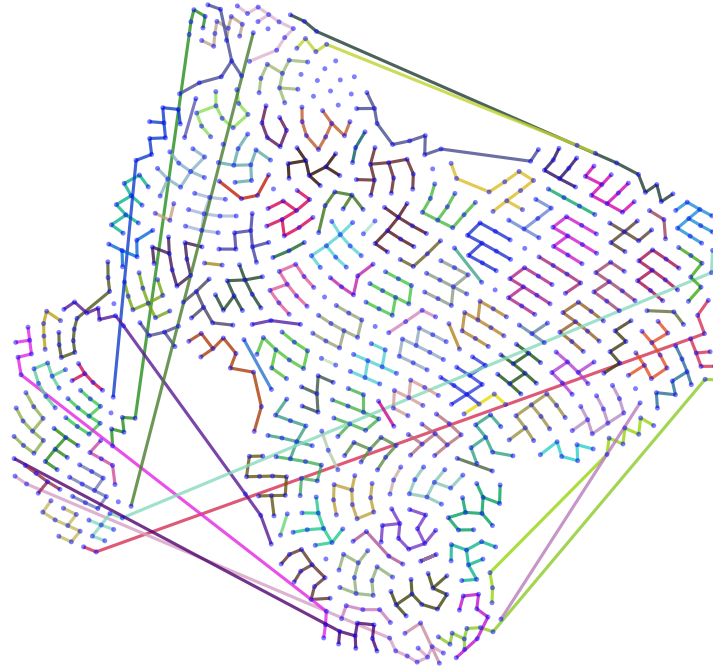


Tapestry Plot of the t-SNE (*perplexity* = 30) scatterplot, with its proximal-arrows selected kNN ( $k = 10$ ) and filtered by DNR ( $\phi = 2$ ). Delaunay edges fill in the gaps within the patches in the scatterplot, making it into a uniform mesh underneath. The visual evidence of one point inside the patch “safely” traveling to another is very clear. The tear-up distortions shown by long red proximal-arrows are not trimmed, though they signal important similarity information, they cover up some parts of the visualisation.)





Tapestry Plot of the t-SNE (*perplexity* = 30) scatterplot, with its proximal-arrows selected kNN ( $k = 10$ ), filtered by DNR ( $\phi = 2$ ) and bundled. With the similar long proximal-arrows bundled, they cover up less space as they do in the previous tapestry plot. With sufficient amount of high rank proximal-arrows, this tapestry plot gives us convincing visual evidences that the 3D dataset is a closed-up 2D manifold, which is the essence of the spherical shell.



Faithful Proximity Forest (FPF) of t-SNE (*perplexity* = 30) scatterplot with the input space neighbourhood estimation scheme as kNN ( $k = 30$ ). The points connected by each minimum spanning tree is a mutual proximity closure, an intersection of their input space neighbourhoods, with the average size around 10. With the visual cues suggested by the previous tapestry plot, we know these small closures could join together to form a bigger manifold.

## 6.4 Visualisation Sequences of Earth Cities

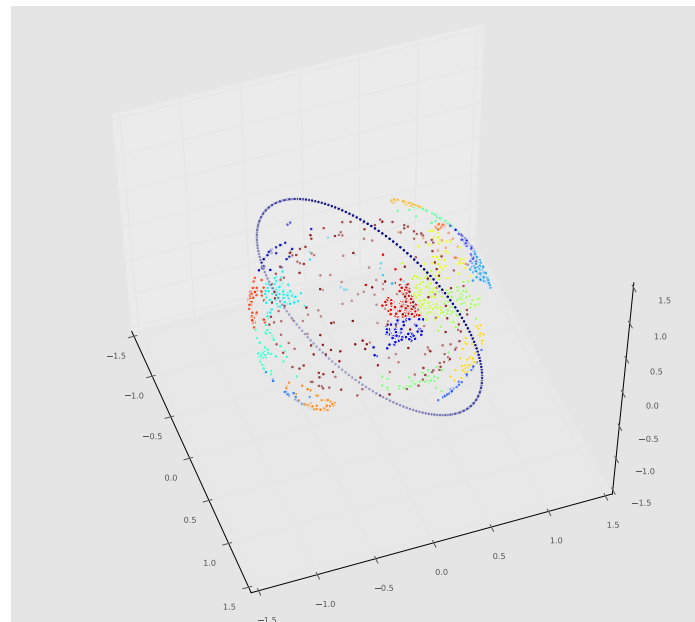
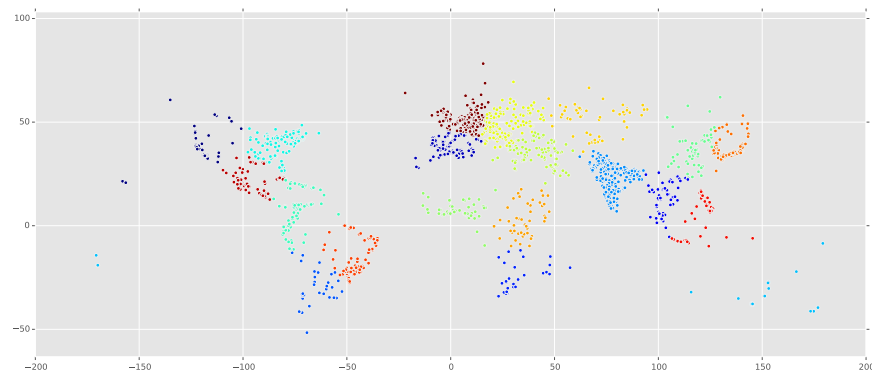
This section presents a new showcase dataset called Earth Cities, it is like a 3D spherical shell but the points on the 3D spherical surface are in the distribution of the earth cities (2000 randomly sampled) provided by a source available in public domain [MaxMind \(2015\)](#). The radius to the centre of the earth is normalised to 1.0. Also there are 200 points of Gaussian noises inside the earth with their distance to the centre of the earth being 0.8 at most, which can be seen as the “underground cities”. Finally there are 200 points out of the earth surface forming circle with their radius all being 1.2, which can be seen as the “satellites”. The purpose

of this dataset is to demonstrate the capabilities of overlay graphs to explore an organic dataset with noises and other distinguishing structures.

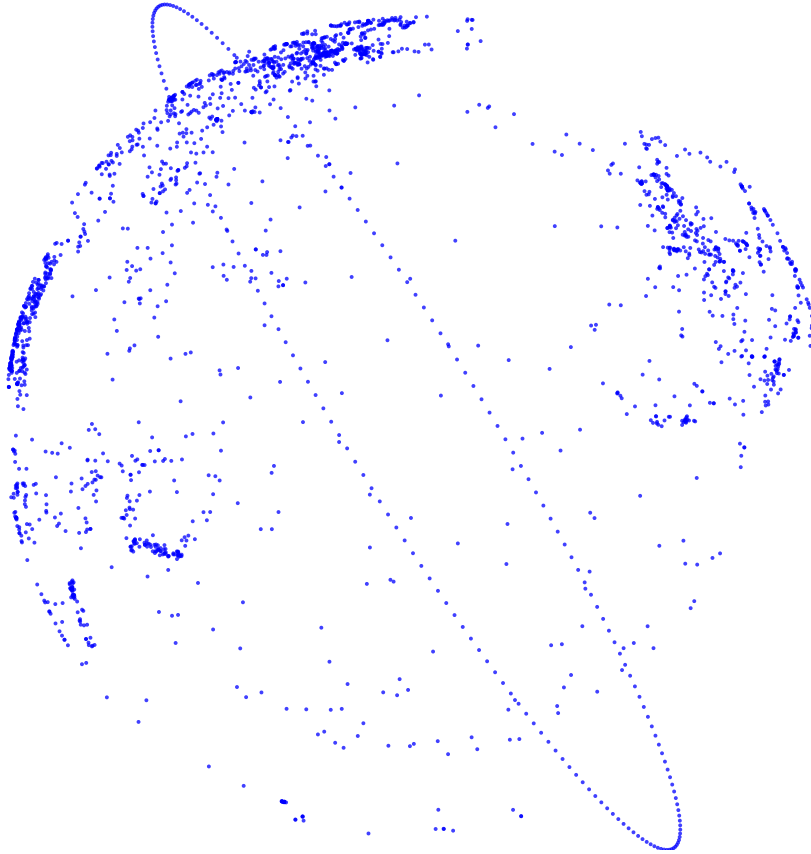
The visualisation sequences are provided as following:

- The world cities presented in the usual map in 2D space without underground cities and satellites.
- Earth cities dataset fully presented in 3D space.
- PCA scatterplot.
- t-SNE scatterplot.
- Delaunay plot of PCA scatterplot.
- Delaunay plot of t-SNE scatterplot.
- $\lambda$ -Proxigram of t-SNE visualisation.
- Tapestry plot of t-SNE visualisation.
- Faithful Delaunay patches map of t-SNE visualisation.
- Faithful Proximity forest of t-SNE visualisation.
- PCA and t-SNE scatterplot with data points label colours.

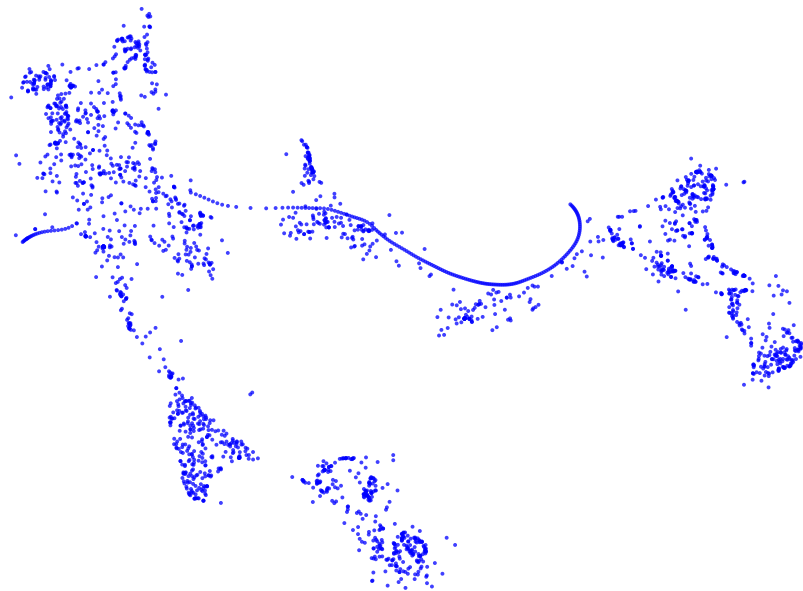
The dataset has 2400 points in total. In order to colour the points, the earth cities are categorised in to 20 “countries” by k-nearest neighbours clustering. The input neighbourhood estimation scheme is stochastic neighbours ( $perplexity = 30, w = 2$ ).



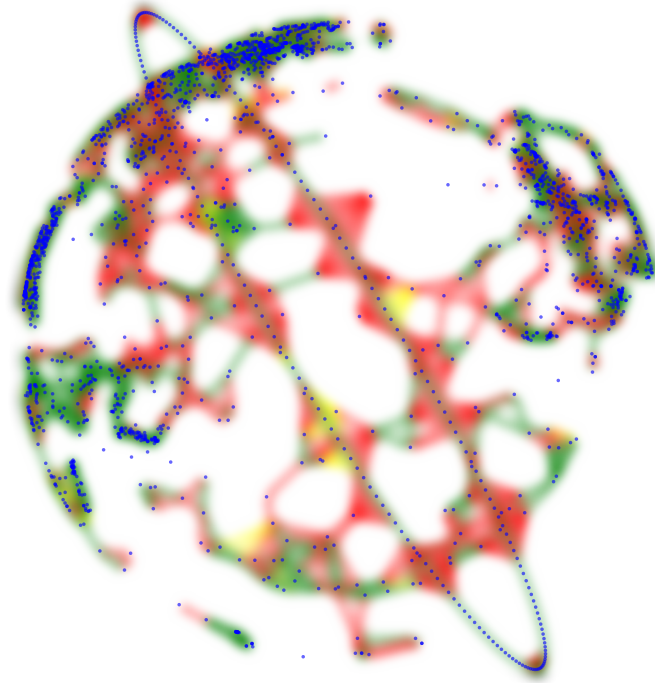
The upper diagram is the 2000 earth cities only laid in the usual world map view with their label colours showing 20 clusters. The lower diagram is the whole earth cities is drawn in 3D space. The dark blue are the satellites and the dark red are the underground cities. This is the dataset PCA and t-SNE is going to process.



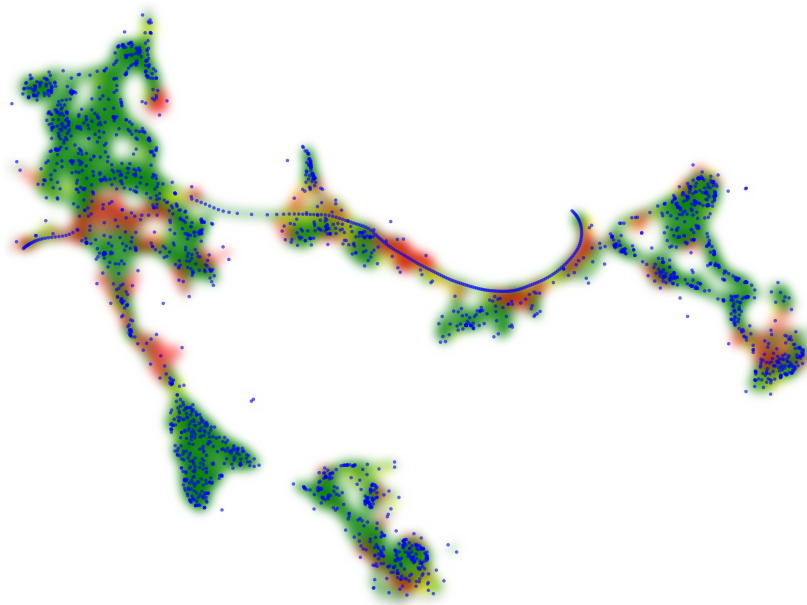
The plain PCA scatterplot visualisation. The two principal components clearly capture the direction of the maximum variance, as the cities points are clashed into the two edges of the sphere. The structure of the satellites are clearly seen, so are the underground cities. However the cities of two sides of the sphere are mixed together in the diagram, which results in massive false-neighbours in visualisation quality standard.



The plain t-SNE (*perplexity* = 200) scatterplot visualisation. Unlike PCA, this diagram does not look like the earth in any way, because t-SNE preserve local structure rather than global structure. From the diagram, we can see some big patches, a curve-line across the patches and only a few outliers.

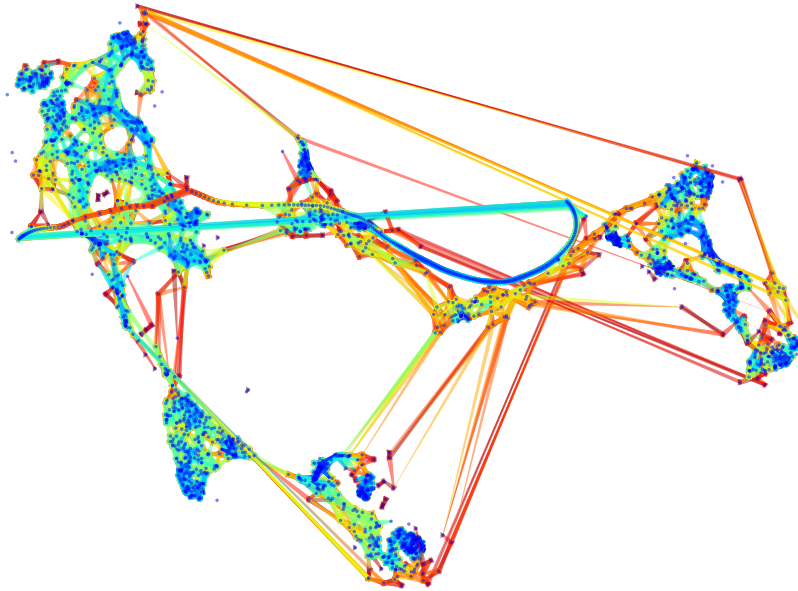


The Delaunay plot of the PCA scatterplot. The middle of the diagram is sparse, thus some relatively long Delaunay edges are uncut. But many Delaunay edges in the middle are red, which means they are false-neighbours. We can foresee that this is caused by overlapping two sides of the sphere, which lets some cities, underground cities and the satellites be mixed together. There are some patches where the green Delaunay edges are the majority but with a few red Delaunay edges as well. This is the main problem of the diagram because most of the earth cities are smashed into tight clusters which is not ideal for observing the local structures. Lastly, the Delaunay plot Cleanliness is low as 0.757 with 1356 false-neighbours revealed. We will stop doing further overlay graphs for this PCA scatterplot from this point.

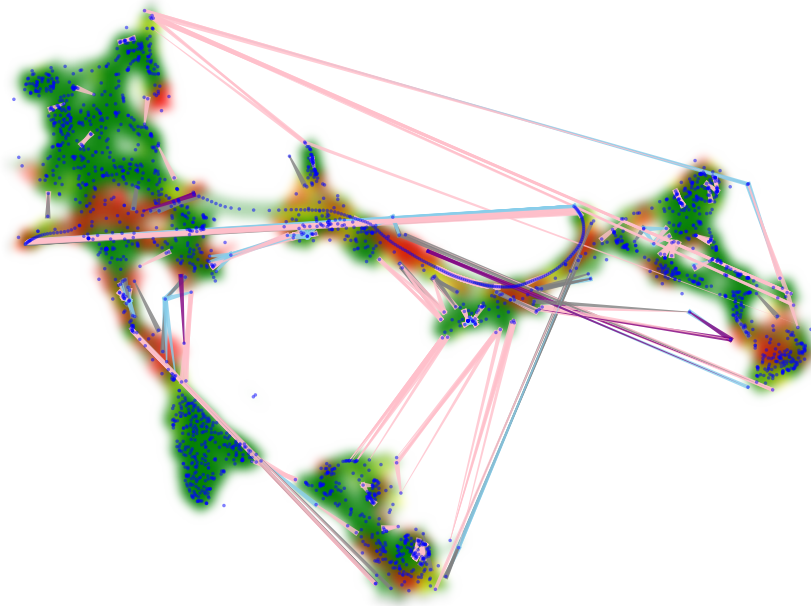


The Delaunay plot of t-SNE scatterplot visualisation. Although the visualisation does not reflect global geometry of the data, it does preserve local structure better than PCA does. Most of the Delaunay edges in the patches we saw in the plain t-SNE scatterplot are green, which means the points are placed closely to their true neighbours in the 3D space. While some parts of the green patches have red Delaunay edges embedded. But without Proxigram we do not know which point it should be close to in the 3D space. Finally Cleanness measure of this Delaunay plot is high as 0.938 with 369 false-neighbour revealed.



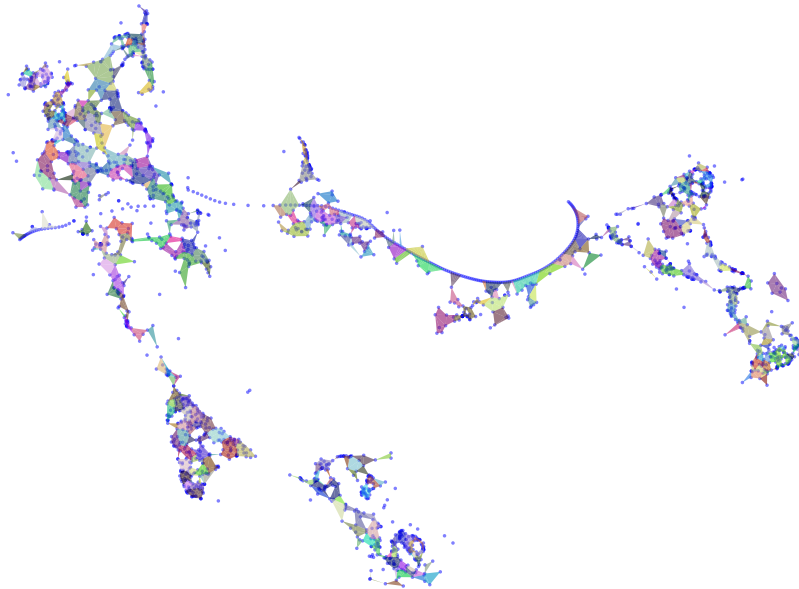


The  $\lambda$ -Proxigram ( $\lambda = 0.8$ ) of t-SNE scatterplot visualisation. There are 41047 proxi-arrows in the diagram. Clearly we can see some points in the visualisation do not have any proxi-arrows. They are outliers identified by stochastic neighbours. The colour of the proxi-arrows are coded by their level of proximity. The long red proxi-arrows are the worst tear-ups because high level proximity relationships are distorted. While there are short blue proxi-arrows are the majority, even the neighbours of low level proximity are placed fairly closely by t-SNE.

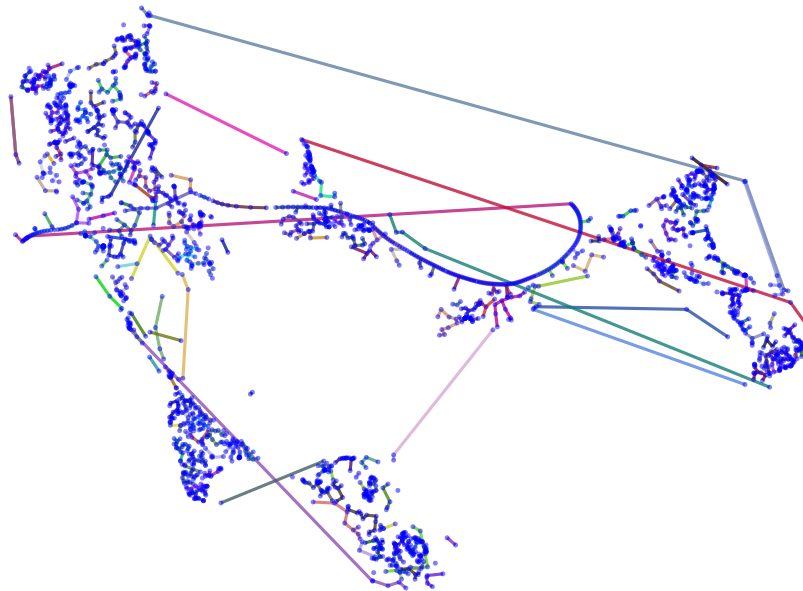


The Tapestry plot of t-SNE scatterplot visualisation, with its proximal arrows are filtered by Delaunay neighbours ( $\phi = 5$ ). There are 613 tear-up proximal arrows in total, with their tear-up type colour scheme as: joint – *pink*, subtle – *grey*, *false-outlier* – sky blue and *false-neighbours* – purple.

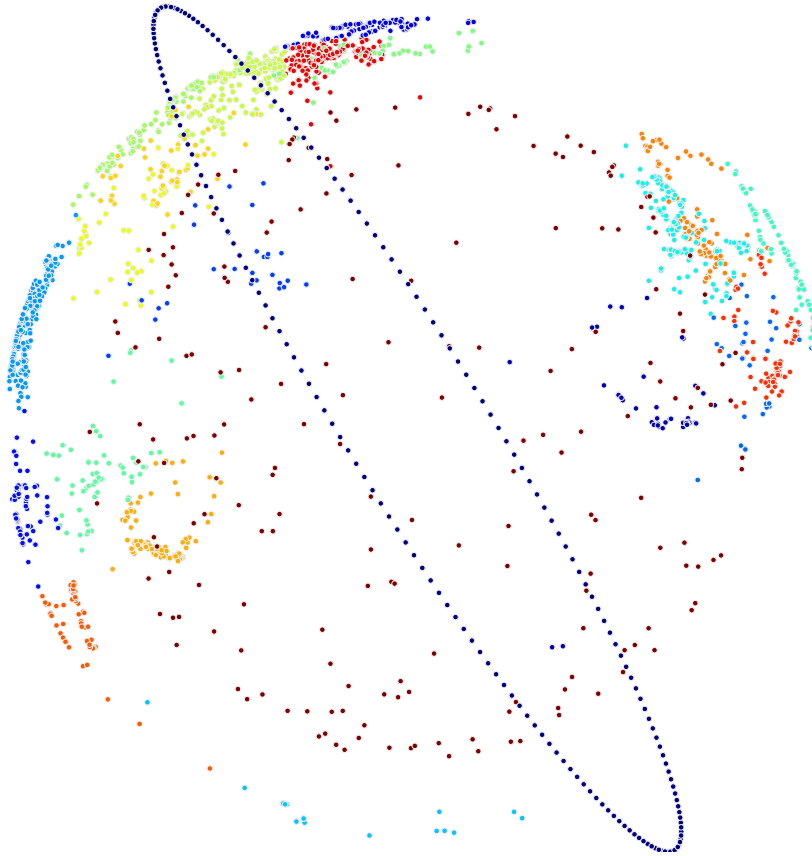
Most of the tear-up proximal arrows are pink, which implies two things: firstly those source-points are visualised comfortably with their nearby neighbours in the 2D space; Secondly the small scale local structures are split by t-SNE. The few existed purple proximal arrows show the true neighbours to those source points, which are clearly false-neighbours to their surrounding points in the 2D space. The grey tear-up proximal arrows suggests that some outliers are not real because t-SNE fails to put those points closely to their true neighbours. Lastly, there are some points embedded with red Delaunay edges have no proximal arrows. These are the worst false-neighbours (inliers) because they should be put isolated as outlier in the visualisation but t-SNE fails so.



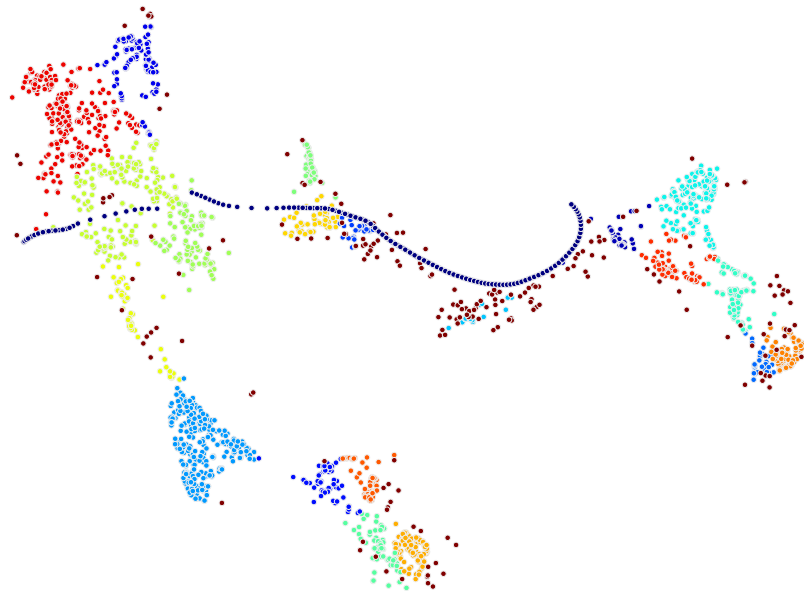
The faithful Delaunay patches map (FDPs) of t-SNE scatterplot visualisation with input neighbourhoods estimation scheme as kNN ( $k = 50$ ). The FDPs is a diagram showing how well the mutual proximity relationships the scatterplot could have. With red Delaunay edges not included, the previous green meshes are replaced by the smaller faithful Delaunay patches adjacent to each other with holes, which reflects how the earth cities are sampled.



The faithful proximity forest (FPF) of t-SNE scatterplot visualisation with input neighbourhoods estimation scheme as kNN ( $k = 50$ ). FPF shows the mutual proximity relationships should be in the original space, it does reveal the tear-up proxi-arrows which are among the proxi-arrows of the highest level of proximity. Most of the spanning trees are curly small which means the local structures of the majority are visualised well. The critical tear-up in the spanning trees suggest the splitting in these small mutual proximity closures by t-SNE.



The scatterplot of PCA with data point label colour. As we expected, the visualisation has the two sides of the sphere overlapped. The number of false-neighbours is not as many as in 3D spherical shell dataset is because the overlapping space is in fact the oceans on the earth. The underground cities are indeed mixed with earth cities regardless the fact that they lie on a different manifold.

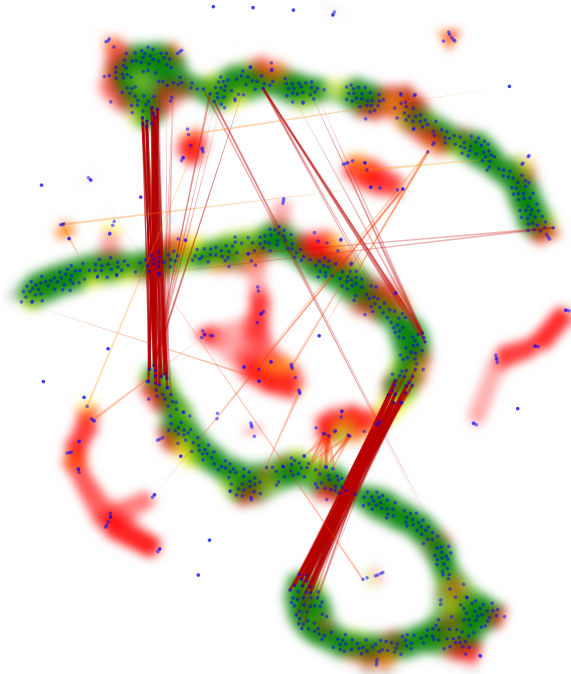


The scatterplot of t-SNE with data point label colour. After several overlay graph we finally see the true face of the t-SNE visualisation. The colour gradient in the big patches are continuous which means the local structures are preserved well. Some “countries” are split into two, which tearup-Proxigram has suggested. The satellites is like a snake passing through the earth and underground cities, which Delaunay plot has shown the false-neighbour relationships they initiate. Some underground cities are placed within some “countries”, which Delaunay Plot has given the visual evidences.

## 6.5 Visualisation Sequences of Snakes Dataset

This artificially generated dataset has 1000 data points. 800 of these are from a single winding cluster like a snake; the other 200 are outliers distributed as Gaussian noise surrounding the snake. Such data resembles a multidimensional time series with random outliers. The below diagram is the tapestry plot of t-SNE (*perplexity* = 30) scatterplot of this snake dataset. The input neighbourhood estimation scheme is stochastic neighbours (*perplexity* = 25,  $w = 3$ ) and the level of truth factor for the tearup-Proxigram is  $\lambda = 1.0$  (complete estimation).

The t-SNE optimisation achieves only a local minimum, for the snake is cut into sections, which should be joined but are not. In the scatterplot alone we can not see there is only one cluster. This Tapestry plot shows which section is connected to which, and also shows which points are in fact outliers, even they are mistakenly placed closely to the clusters by t-SNE. Thus we can conclude that the Tapestry plot helps to successfully reveal the entire snake structure and identify the outliers.



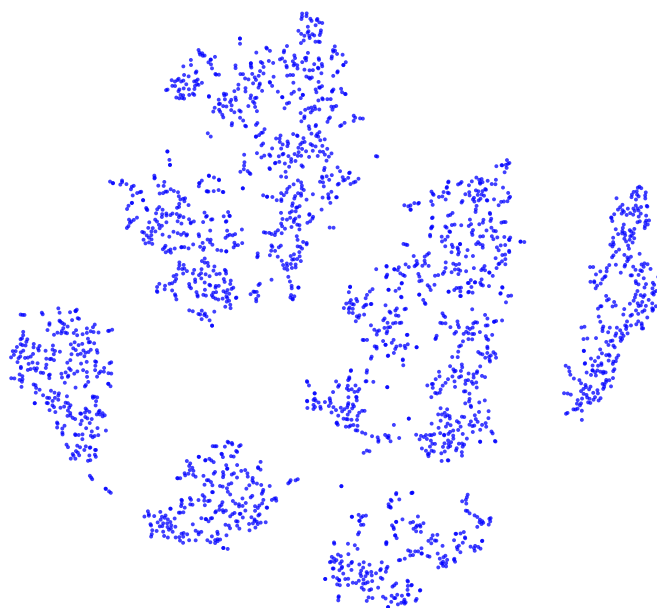
## 6.6 Visualisation Sequences of MNIST

This section presents a showcase of using a variety of overlay graphs and t-SNE to explore the underlying structure of the ten digits in the MNIST [LeCun et al. \(1998\)](#). The test data is a 2500 random sample from the whole dataset with 784 dimensions. MNIST dataset was used as the showcase in the paper by [Van der Maaten and Hinton \(2008\)](#) to demonstrate the superiority of t-SNE. Thus we only use t-SNE for this dataset as the disadvantages of other DR methods are already shown in the paper of t-SNE. The input neighbourhood estimation scheme is stochastic neighbours ( $perplexity = 45, w = 3$ ).

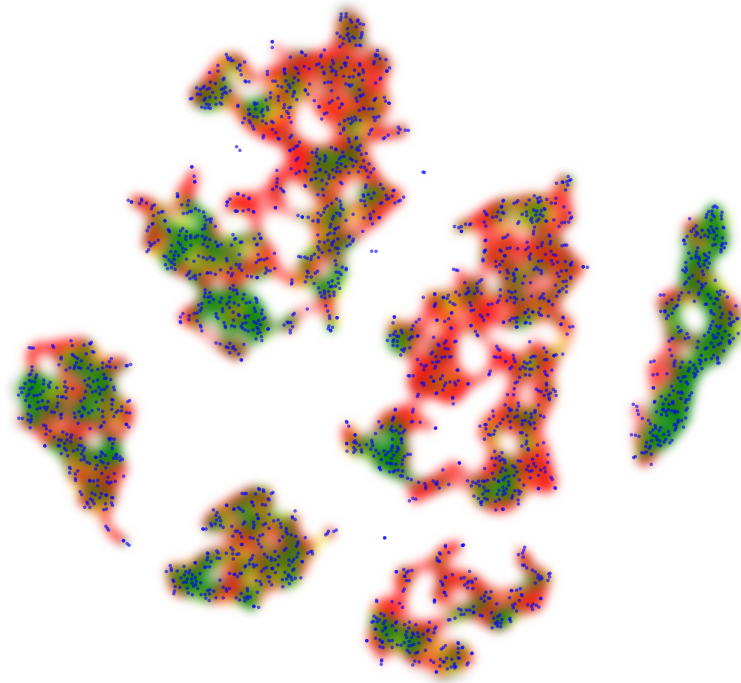
A sequence of visualisation will be presented as following:

1. The scatterplots of t-SNE in plain colour.
2. The Delaunay plot of t-SNE scatterplot.
3. The  $\lambda$ -Proxigram of t-SNE scatterplot.
4. The Tapestry plots.
5. The faithful Delaunay patches map.
6. The faithful proximity forest.
7. The scatterplots of t-SNE with label colours.

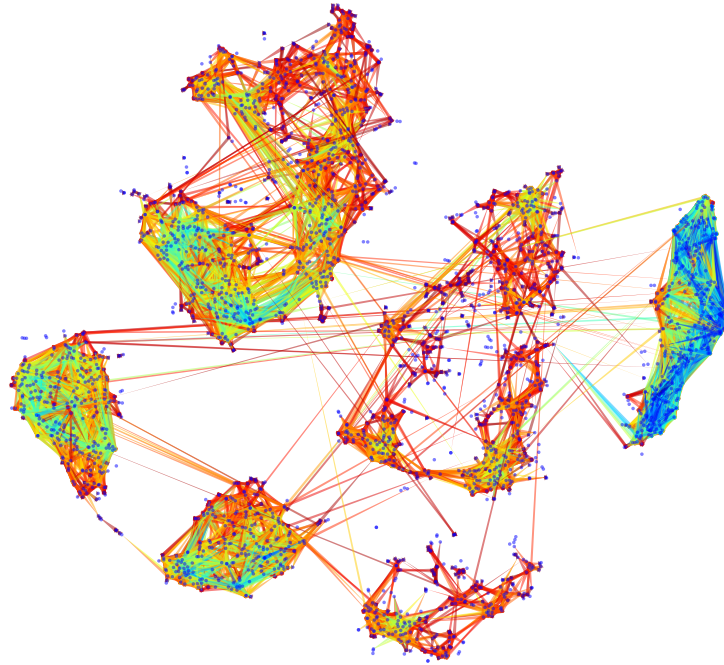




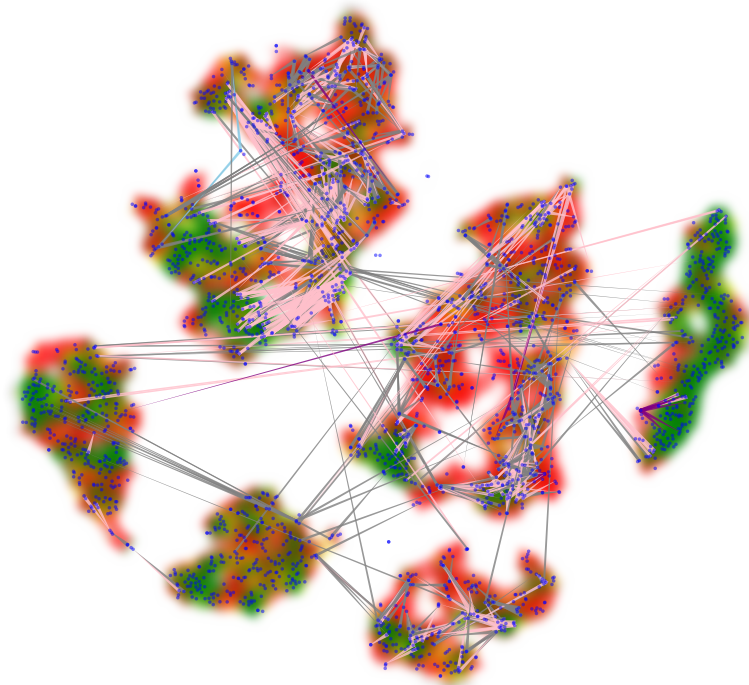
The t-SNE scatterplot of 2500 MNIST data points in plain colour. The DR method remarkably visualises the complex real world dataset of high dimensionality. But without the label colours can we successfully recover the similarity information of the data via overlay graphs?



The Delaunay Plot of t-SNE scatterplot on MNIST dataset, with the input neighbourhood estimation scheme as stochastic neighbours ( $perplexity = 45, w = 3.0$ ). The scatterplot roughly has 6 large clusters or 11 small clusters depend on subjective term. However there are 5 such clusters have sufficient amount of green Delaunay edges with few red Delaunay edge embedded in. The cleanness of Delaunay plot is 0.690 with 2023 revealed “false-neighbours”. So we know some local structures are confirmed by Delaunay plot to be visualised well.

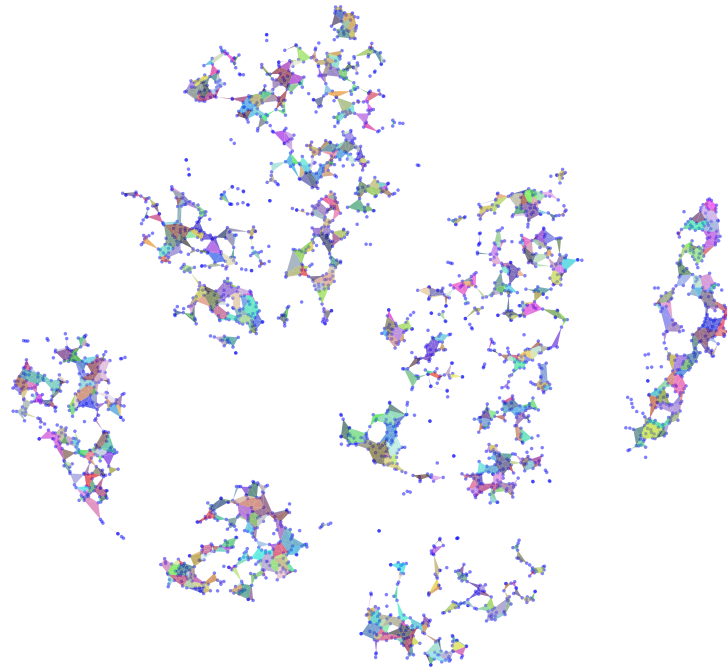


$\lambda$ -Proxigram of t-SNE scatterplot on MNIST. The general Proxigram has massive amount of proxi-arrows that can not put in the diagram, I choose the level of truth factor as  $\lambda = 0.7$ . The proxi-arrows natural densities give us a general idea of structure of clusters. The small clusters are covered by considerate amount of short proxi-arrows (green and blue), which implies the clusters might possess mutual proximity relationships. The large or sparse clusters have fine structures as the proxi-arrows within mostly crossing in between.

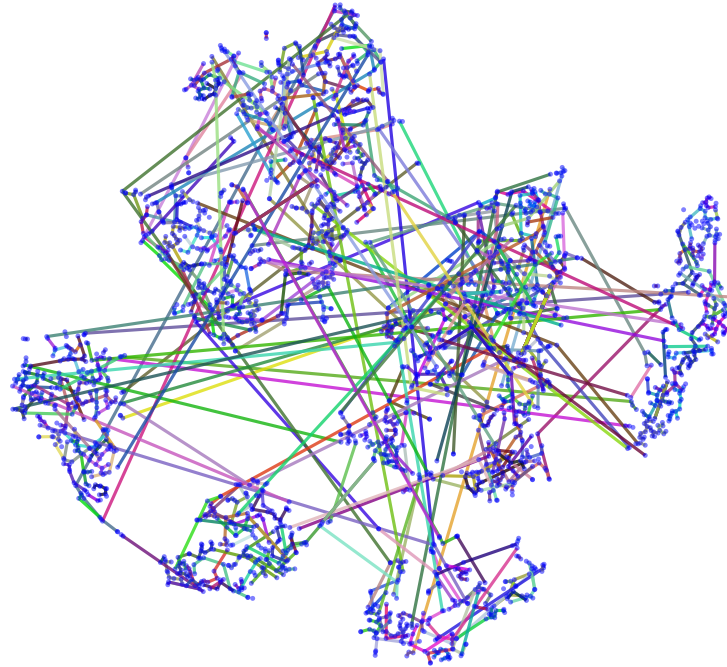


The Tapestry plot of t-SNE scatterplot on MNIST, with its proxi-arrows from the  $\lambda$ -Proxigram ( $\lambda = 0.7$ ), filtered by Delaunay neighbour reachable with threshold set to its upper limit. The adjustment of using extreme DNR is because the small clusters are already isolated and the sparse clusters embedded with red Delaunay edges. Expanding the output neighbourhoods to their limit helps to reduce the small scale proxi-arrows within small clusters and reveal the overall distortion of the scattered sparse clusters.

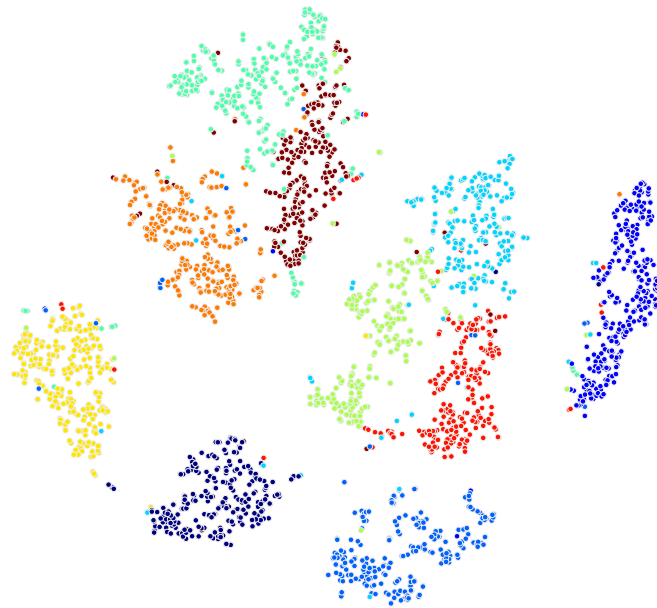
From the selected tear-up proxi-arrows in the Tapestry plot, we can see the majority of them are joint tear-up (pink colour), especially in the upper left part of the visualisation, because there are no green paths in between those split structures to dissolve their tear-up proxi-arrows. the local minimal nature of t-SNE optimisation function falls to piecing theses fine structures together, but the overlay graph compensate the missing information. The outskirts points attached to the cluster of green Delaunay edges are completely misplaced but the scatterplot produced by t-SNE gives the illusion that they are part of the attached clusters.



The faithful Delaunay patches map of t-SNE scatterplot on MNIST, with its input neighbourhood estimation scheme as k-Nearest neighbours ( $k = 50$ ). The diagram shows the mutual proximity closures are scattered as the most faithful Delaunay patches are small and disconnected. The right part of the scatterplot is an exceptions, which suggests it is indeed a tight cluster.



The faithful proximity forest of t-SNE scatterplot on MNIST, with its input neighbourhood estimation scheme as k-Nearest neighbours ( $k = 50$ ). The mutual proximity closures shown by the minimum spanning forests has too many crossing tear-up proxi-arrows (seems to be more than Tapestry plot, but it is hard to see). The diagram suggests the greedy algorithm finds many mutual proximity closures and some of them has serious tear-up distortions.



The t-SNE scatterplot with label colours. One can say t-SNE has most of the points placed reasonably well however it is not enough, because clearly some points attach to some strange clusters with different label. The two sparse clusters are the mixed of three small clusters, which without the colour labels are very misleading. However from observing the overlay graph, we have successfully discover the inliers, cluster density and splitting structures.

Without overlay graphs we are unable to recover detailed HD similarities information, tear-up and false-neighbour distortion in the plain t-SNE scatterplot. When the digit labels are not available, even the cutting edge dimensionality reduction method could lead to wrong interpretation of the data.

## 6.7 Visualisation Sequences of Glass Dataset

This section presents a showcase of overlay graph visualisation to a well-known real-world dataset — Glass Identification. The data is available in the UCI Machine Learning Repository website for classification problem and was originally discussed in [Evelt and Spiehler \(1987\)](#), which is cited at least 59 times since then. It consists of 214 instances, each is a glass fragment with 9 numeric measurements. The first measurement is refractive index (RI) and the rest 8 measurements are the weight percentages of elements in corresponding oxide. The elements are Sodium (Na), Magnesium (Mg), Aluminium (Al), Silicon (Si), Potassium (K), Calcium (Ca), Barium (Ba) and Iron (Fe). The data has 6 classes and it can be decided into 2 broad categories — window glass and non-window glass. The types of glasses are summarised as following Table 6.1.

Class	Code	Category	Instance
Float Processed Building Window	#0	Window (total 163)	70
Non-Float Processed Building Window	#1		76
Float Processed Vehicle Window	#2		17
Container	#3	Non-Window (total 51)	13
Tableware	#4		9
Headlamps	#5		29

Table 6.1 Glass Identification Class Information

The initial study [Evelt and Spiehler \(1987\)](#) suggests that the classification performances of the  $k$ -nearest neighbour method, discriminant analysis and their own rule-based method BEAGLE are not satisfying especially dealing with the classes from the non-window category; But the results are relatively better when classifying the data into two border categories — window and non-window glasses. There could be more sophisticated classifying method for Glass data, however it is beyond the scope of this thesis. I will only use the original features for the later visualisation sequence as they were in the initial study.

Applying overlay graphs requires good quality for the HD data scatterplot, otherwise the diagrams become too complicated to read. I will use the global quality measures *Nearest Neighbour Correctness Rate* and *Venna's Trustworthy and Continuity* to search the best perplexity for t-SNE. The distance rank parameter  $k$  for T & C is restricted to 5, as the classification difficulty mentioned in [Evelt and Spiehler \(1987\)](#).



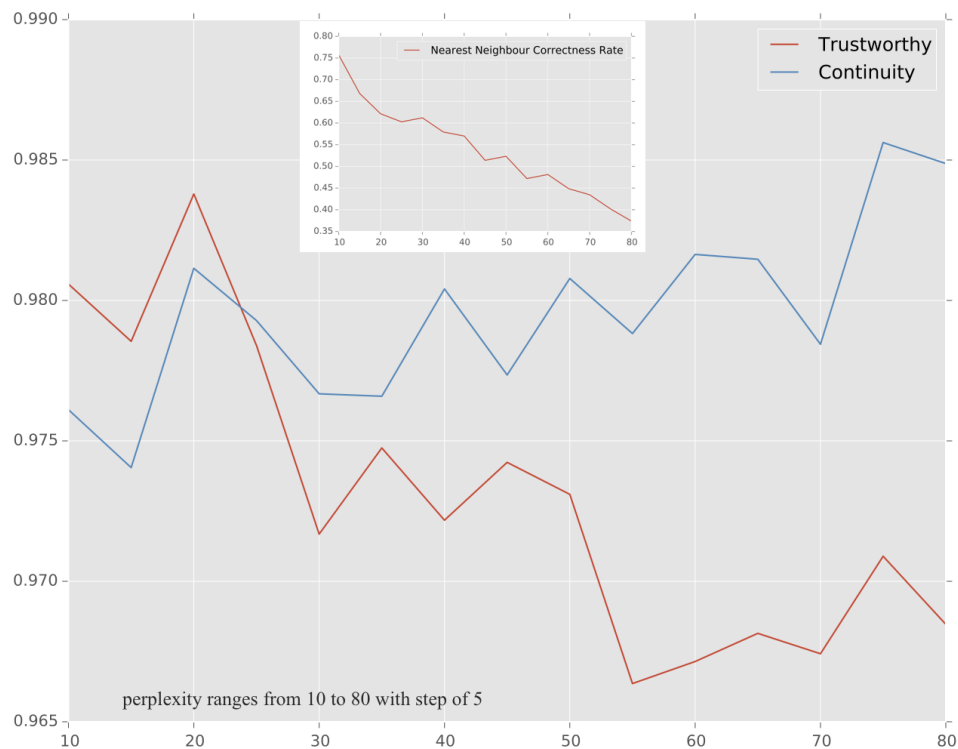


Fig. 6.1 Global Qualities for t-SNE on Glass Data

Fig. 6.1 the plot of the global qualities measures for the t-SNE visualisation to the Glass Identification dataset with the perplexity of t-SNE ranging from 10 to 80 with a step of 5. Since we want to ensure the good local proximity quality, perplexity = 20 appears to be the best choice in terms of both nearest neighbour correctness rate and Venna’s T & C.

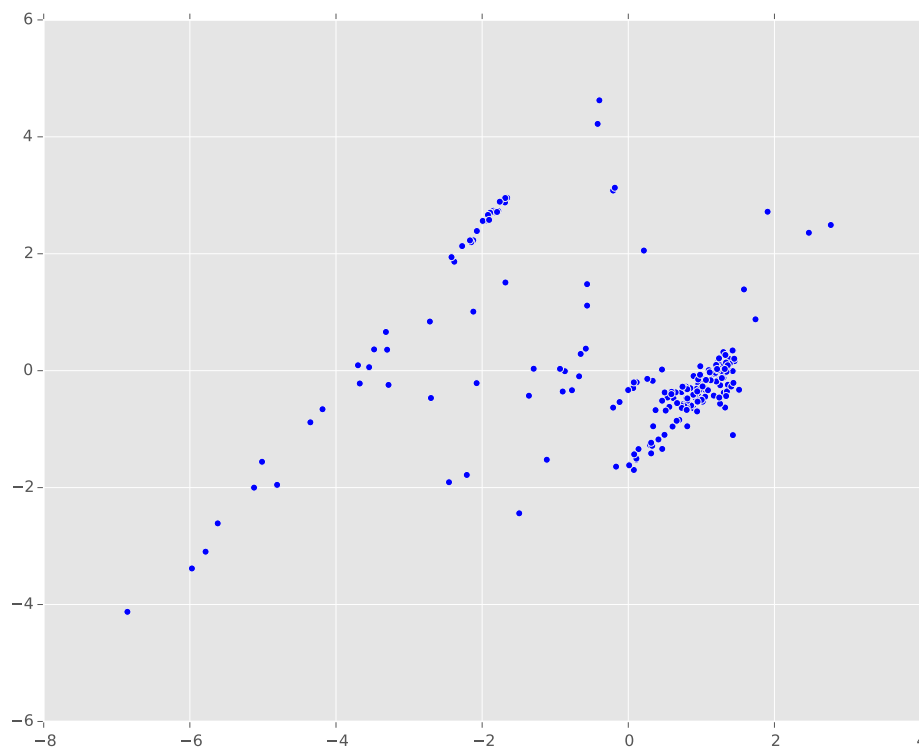


Fig. 6.2 PCA Visualisation of Glass Dataset

The visualisation results from PCA (shown in Fig 6.2) has a very low nearest neighbour correctness rate (0.17) and suffer from the crowding problem, so the do other traditional DR methods reviewed in Chapter 2, thus they are excluded in further discussion.

At first we will use a sequence of visualisations to probe the suitable overlay graph settings. Then I will present the scatterplot coloured with the class labels and overlaid with the tapestry plot to explain the nature of the dataset.

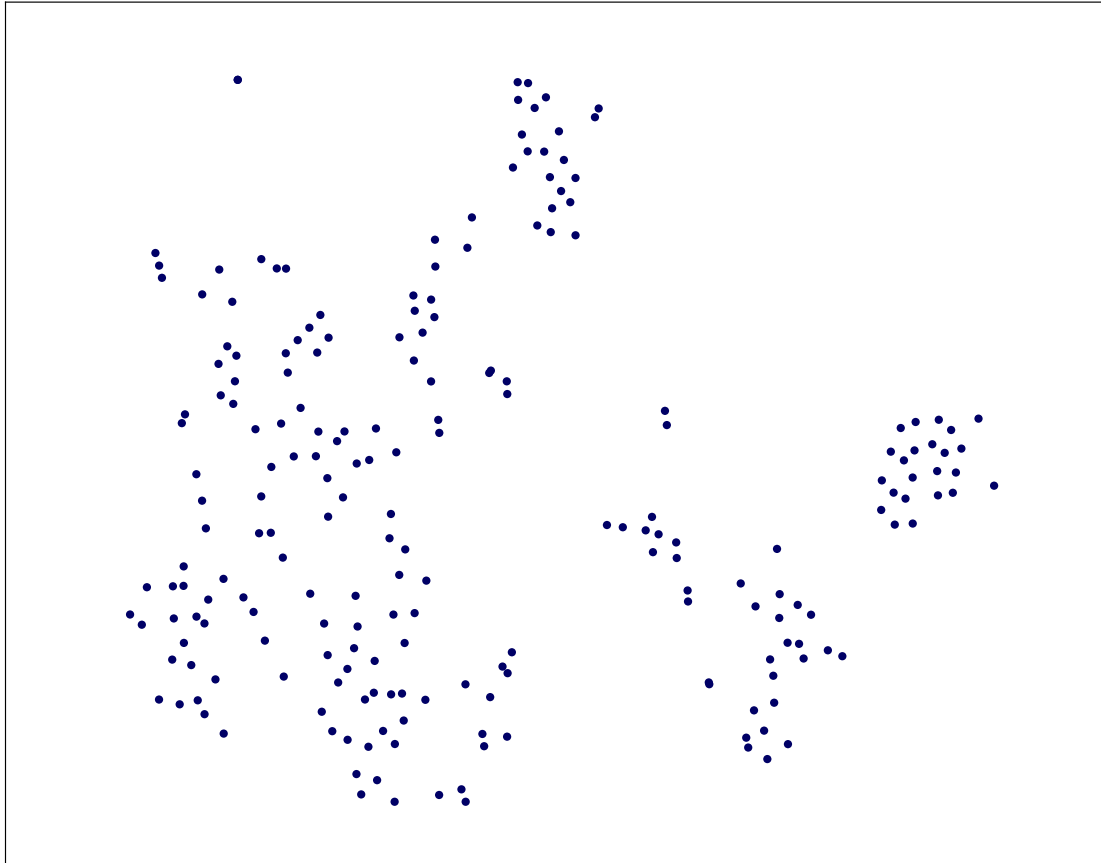


Fig. 6.3 t-SNE (perplexity = 20) Visualisation of Glass Dataset

Fig. 6.3 is the plain scatterplot produced by t-SNE with perplexity = 20. As the classification problem for this dataset is difficult, we do not expect the clusters in the diagrams to be the glass classes. There are 3 small clusters and 1 big cluster, although some of them are not completely separated. There are also a few obvious outliers in the diagram, but without Delaunay plot and proxigram, we can never trust whether they are as they appear to be.

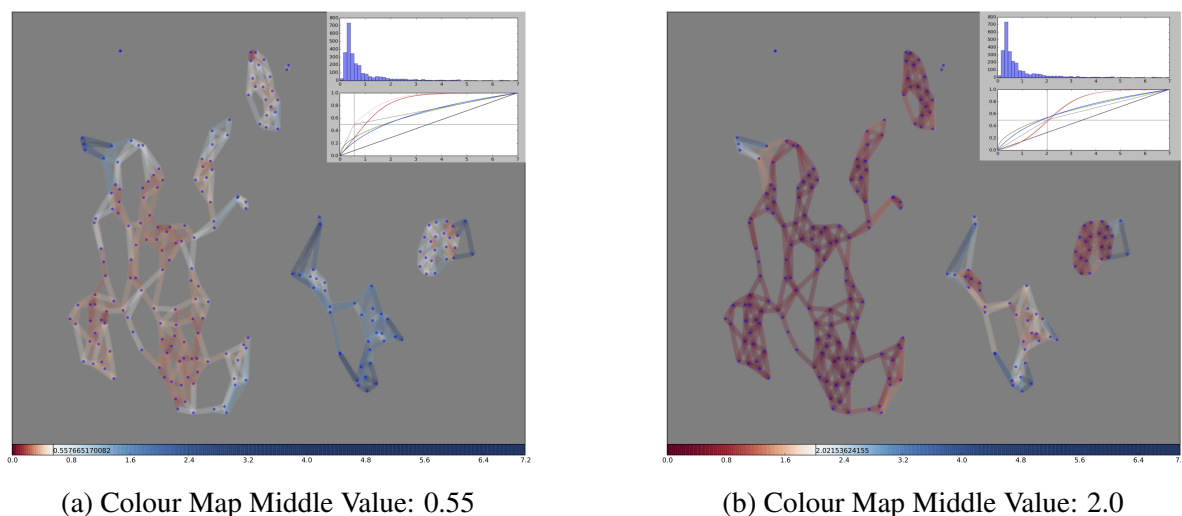


Fig. 6.4 Delaunay Plots of t-SNE Visualisation Coloured in HD Distance

The method used to cut the long Delaunay edges is Stochastic Neighbour, with perplexity = 20 and  $s = 2.5$ . As shown in Fig 6.4, the cutting scheme has removed the long edges fairly effectively. The HD distances are used to colour each Delaunay edge using sigmoid function. A diverging colour map called *BuRd Brewer* (2006) is used, so that the similar pairs in the HD space appear as red colour and dissimilar pairs are in blue. The *middle value* of the colour map is the key point for the Delaunay plots exploration, as it raises from low to high, the colour pattern of the edges changes. It is difficult to deliver the interactive experience here thus I will only give two diagrams to demonstrate how the middle value help to discover structures in the Delaunay plot.

The middle value of the colour map in Fig. 6.4a is fairly low as 0.55 according to the histogram (top right corner of the diagram) of the HD distance distribution of all the Delaunay edges presented. The red Delaunay edges in the sparse cluster on the left suggest that this cluster is much more densely distributed than it appears to be in the scatterplot. It also tells that the points in this sparse cluster are more closely together than the other three in the HD space.

As the middle value of the colour map in Fig. 6.4b raises to 2.0, the edges in some clusters become all red except the bottom right one, within which some blue edges are shorter than the red and white ones. They suggest that the proximities in the scatterplot are not reliable to reflect the similarities in the HD space. The inliers attached to the outskirts of some clusters become obvious to find as their Delaunay edge colours are still blue even the middle value increases.

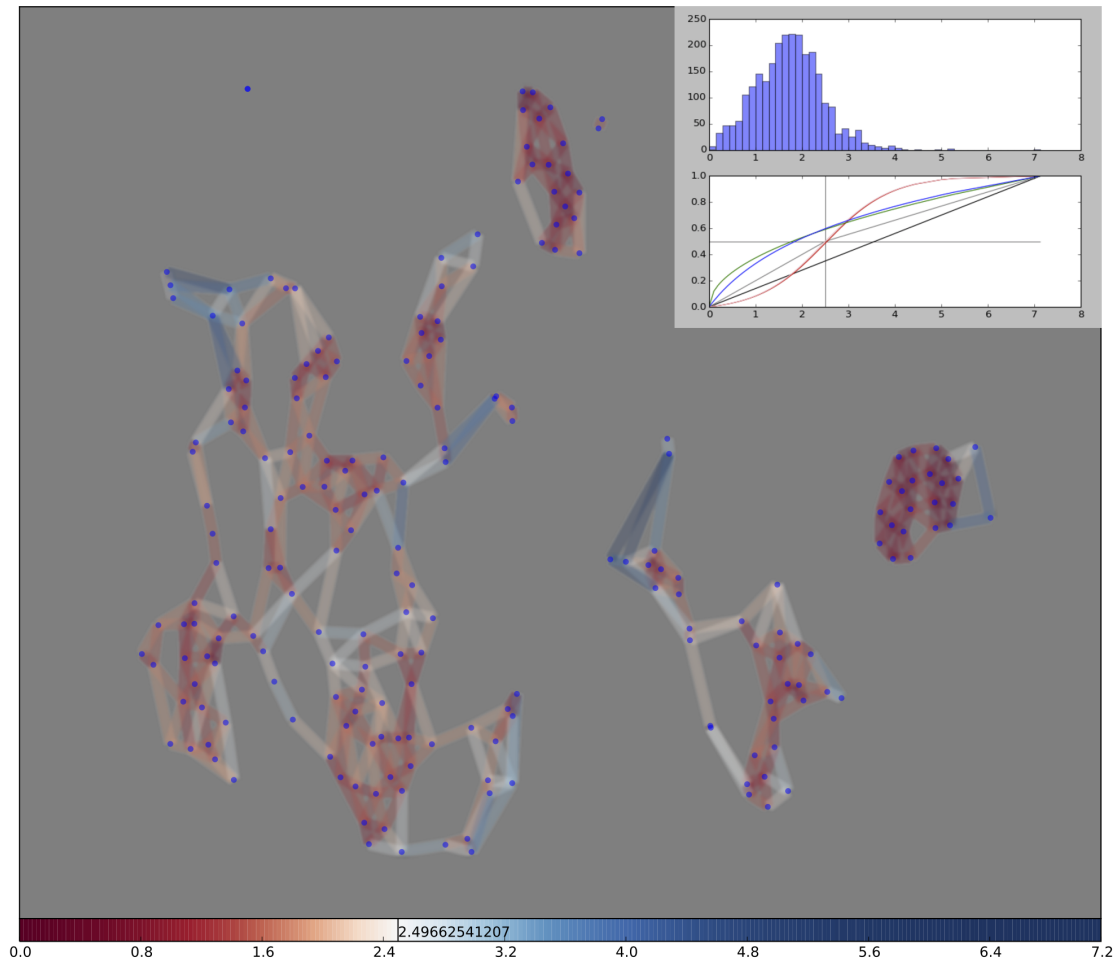


Fig. 6.5 Delaunay Plot of t-SNE Visualisation Coloured in SN score (perplexity = 20)

So far we know how the points in the 4 cluster really distributed in the HD space and where the inliers are. To compensate the scale differences among different clusters so that the colour red only indicate the pair being “similar” in the local sense, I rescale the HD distances by using stochastic neighbour score as the similarity metric (perplexity = 20).

The middle value of the colour map in Fig. 6.5 is set to be 2.5, because the benchmark value for SN is 3.0 but we want the Delaunay edges to have the most diverse colour scale. The new colour scale suggests the 4 clusters and the inliers we suspect in the previous two Delaunay plots are credible. However the big cluster on the left of the scatterplot could be a mixture of different structures as some edges within the mesh of red edges are in white and pale blue colour.

As Venna's T & C suggests that the t-SNE scatterplot has a good global quality when  $k = 5$ , I will use it as a guidance for the setting of proxigram. Fig. 6.6 is the 5-nearest neighbour proxigram to the Glass data visualisation with the same colour scale as in Fig. 6.5. The deep red prox-i-arrows in the diagram show the most dense parts within the 4 clusters in the HD space. There are also a few long prox-i-arrows, some of which are in pale-red colour and some are blue, which indicate tear-ups and true outliers respectively.

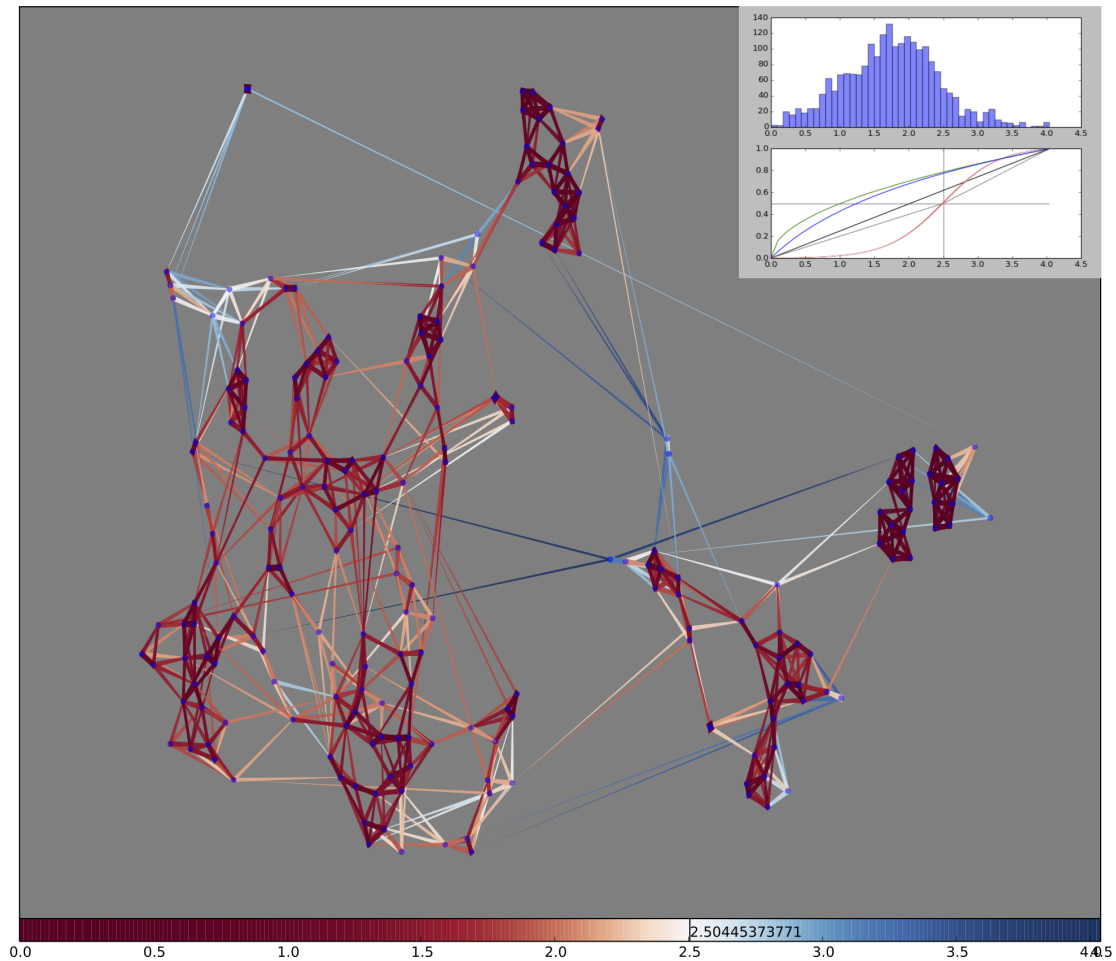


Fig. 6.6 Proxigram ( $k = 5$ ) of t-SNE Visualisation Coloured in SN (perplexity = 20)

There are two types of prox-i-arrows we need to filter in order to produce the tear-up proxigram and tapestry plot:

- the pairs that are both dissimilar in the HD space and far apart in the 2D space (long deep blue prox-i-arrows);
- and the pairs that are both similar in the HD space but already very close in the 2D space (short deep red prox-i-arrows).

The setting of the tear-up proxigram are listed in Table 6.2. The selection scheme has two parts, *primary* and *secondary*, it means the proxi-arrows are only selected to draw when the HD distance rank of the pair is within 5 and its stochastic neighbour score is within 3.0. The secondary scheme is necessary in dealing with this real-world dataset because from the histogram in Fig. 6.6 we know the stochastic neighbour scores of some pairs selected by KNN method are above 3.0, which are unlikely to represent “similar” in the HD space.

	Primary	Threshold	Secondary	Threshold
<b>Selection Scheme</b>	KNN	$k = 5$	SN (perplexity = 20)	$s = 3.0$
<b>Filtering Scheme</b>	DN	$k = 1$	DNR (discrete SN score (perplexity = 20; $s = 3.0$ ))	$k = 3$

Table 6.2 Settings of Tear-up Proxigram

The primary method for the filtering scheme is essential because we do not need those proxi-arrows overlapping with the Delaunay edges; While the threshold of its secondary method is found by interactively adjusting in the computer programme until the removal of the short proxi-arrows achieves user’s satisfactory. The threshold ( $k = 4$ ) of DNR allows those short proxi-arrows to be removed, if the pairs are reachable in the Delaunay plot network only via at most 3 edges that are marked as “close” according to their discrete SN scores. As the result of above settings, the number of proxi-arrows in Fig. 6.6 are reduced from 2140 to 134.

Fig. 6.7 is the tapestry plot of the t-SNE visualisation to Glass data with the scatterplot coloured with glass class labels. The colour scale of proxi-arrows and Delaunay edges are the same as in Fig. 6.6. There are a few discovers from this diagram:

- Class #0 has at least two sub types as depicted by the top cluster, the big sparse cluster and the a few in the middle connected by tear-up proxi-arrows.
- The big sparse cluster is a mixture of Class #0, #1 and #2. But the red colours of the proxi-arrows or Delaunay edges of the pairs are generally more intense within their own classes, with a few exception.
- Most of the items in the window category can be separated from the non-window ones. But a few Class #1 samples are more similar to the other category than to the other items of their own class.
- The pattern of Class #3 is not easy to recognisable from the current features due to they are scattered across the diagram, so does the Class #4.
- Class #5 is believed to be easy to classified from others.

These early observations are not only consistent with the findings discussed in [Evetts and Spiehler \(1987\)](#), but also are able to help devise a more sophisticated rule based classifier. The features used in the visualisation algorithm are raw, we can expect the quality of the visualisation to be improved if methods like feature weighting or rescaling are employed, in the case of which the overlay graphs can also be beneficial.

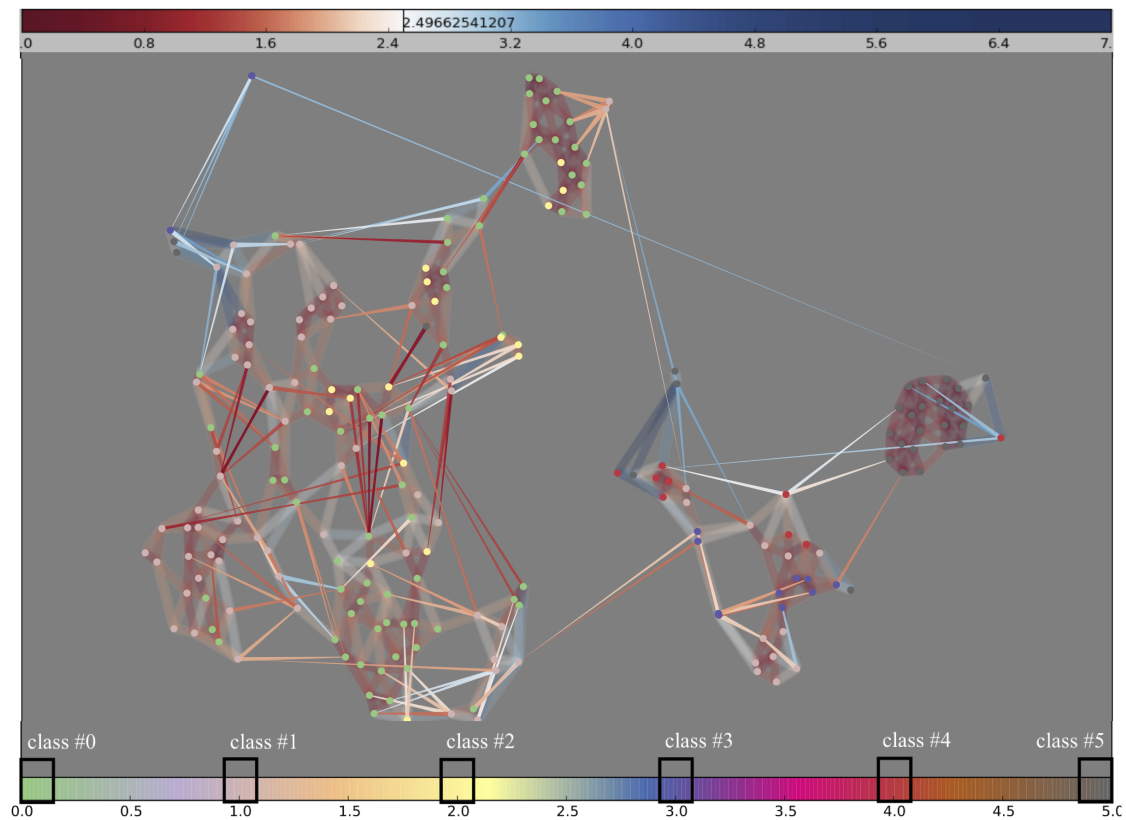


Fig. 6.7 The Tapestry Plot of t-SNE Visualisation



# Chapter 7

## Conclusion

### 7.1 What has been Achieved?

Visualising high-dimensional data as a 2D scatterplot via dimensionality reduction has advanced tremendously. However the traditional HD data visualisation is mainly framed as machine learning problems. It requires the viewer to have knowledge of the specific DR techniques in order to interpret the diagram. Moreover due to limited expressing capability of the 2D space, the scatterplot fundamentally possesses distortions, which blemishes the trustworthiness of the visualisation greatly.

The challenge of the HD data visualisation is not machine learning alone can address, I bring the design aspect to the research and introduce the idea of *Overlay Graph*, an augmentation to the scatterplot by selecting meaningful relationships and displaying them as additional visual components automatically. Throughout the research I have conceived two prime formations of overlay graph, various algorithms of refinements and design choices in the respect of making visualisation both informative and readable. Even some of them are simple, I have not found them in the literature.

The most satisfactory and informative overlay graph is *Tapestry Plot*, a combination of the two prime overlays. As far as I know, this is the only faithful method of informing both tear-up and false-neighbour directly on the scatterplot.

All the works are developed in Python programmes and my interactive application `Project Macaw` is available in public domain, which lays the foundation for the future of high dimensional data visualisation research.

## 7.2 Future Work

The research opens many doors to future work.

1. The concept of HD data overlay graph is in general sense, special overlay graph is possible for application specific purposes or machine learning diagnostic.
2. Once the tapestry plot is drawn, the revealed distortions are not longer counted as errors. The next goal of future algorithm producing 2D embedding of HD data should be a co-optimisation of both reducing errors and improving overlay graph readability. The readability of overlay graph is difficult to measure, but some simple measures such as “data ink” can be used immediately in the new algorithm.
3. So far we interpret the HD similarity from mealy the 2D proximities. The idea of the shortest path on Delaunay triangulation to replace it is also promising. How successful and how much can we retrieve the HD similarity information from the visualisation? This is ultimate question we should ask for the future, if only human knowledge allows to define a visual estimated similarity metric in the 2D space. This requires study to understand how we really read the visualisation.
4. High dimensional data visualisation is a challenge of both our ability to analyse the data and our ability to read the diagram. Fundamental research in psychological perspective that can help us to realise our visual perceptual capacity, memory and expectation for overlay graphs is promising. What are the limits in developing automatic overlay graph for high dimensional data visualisation that human can interpret?

# References

- Althöfer, I., Das, G., Dobkin, D., Joseph, D., and Soares, J. (1993). On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100.
- Balasubramanian, M. and Schwartz, E. L. (2002). The isomap algorithm and topological stability. *Science*, 295(5552):7–7.
- Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.
- Belmonte, N. G. (2013). Multilevel agglomerative edge bundling in javascript.
- Borůvka, O. (1926). O jistém problému minimálním.
- Brewer, C. A. (2006). Basic mapping principles for visualizing cancer data using geographic information systems (gis). *American Journal of Preventive Medicine*, 30(2):S25–S36.
- Bunte, K., Haase, S., Biehl, M., and Villmann, T. (2012). Stochastic neighbor embedding (sne) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, 90:23–45.
- Bushati, N., Smith, J., Briscoe, J., and Watkins, C. (2011). An intuitive graphical visualization technique for the interrogation of transcriptome data. *Nucleic acids research*, page gkr462.
- Chew, P. (1986). There is a planar graph almost as good as the complete graph. In *Proceedings of the second annual symposium on Computational geometry*, pages 169–177. ACM.
- Cox, T. F. and Cox, M. A. (2000). *Multidimensional scaling*. CRC Press.
- De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. C. (2000). *Computational geometry*. Springer.
- Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.
- Demartines, P. and Hérault, J. (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *Neural Networks, IEEE Transactions on*, 8(1):148–154.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

- Evett, I. W. and Spiehler, E. (1987). Rule induction in forensic science. *KBS in Government, Online Publications*, pages 107–118.
- Gansner, E. R., Hu, Y., North, S., and Scheidegger, C. (2011). Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 187–194. IEEE.
- Gonthier, G. (2008). Formal proof—the four-color theorem. *Notices of the AMS*, 55(11):1382–1393.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, 2 edition.
- Heawood, P. J. (1949). Map-colour theorem. *Proceedings of the London Mathematical Society*, 2(1):161–175.
- Hinton, G. E. and Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 833–840.
- Holten, D. and Van Wijk, J. J. (2009). Force-directed edge bundling for graph visualization. In *Computer Graphics Forum*, volume 28, pages 983–990. Wiley Online Library.
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.
- Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lee, D.-T. and Schachter, B. J. (1980). Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242.
- Lee, J., Verleysen, M., et al. (2014). Two key properties of dimensionality reduction methods. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*, pages 163–170. IEEE.
- Lee, J. A., Lendasse, A., Verleysen, M., et al. (2002). Curvilinear distance analysis versus isomap. In *ESANN*, volume 2, pages 185–192.
- Lee, J. A. and Verleysen, M. (2009). Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7):1431–1443.
- Lee, J. A., Verleysen, M., et al. (2008). Rank-based quality assessment of nonlinear dimensionality reduction. In *ESANN*, pages 49–54.
- Lespinats, S. and Aupetit, M. (2011). Checkviz: Sanity check and topological clues for linear and non-linear mappings. In *Computer Graphics Forum*, volume 30, pages 113–125. Wiley Online Library.
- MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

- MaxMind (2015). Free world cities database.
- Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.
- Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.
- Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.
- Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, (5):401–409.
- Saul, L. K. and Roweis, S. T. (2003). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research*, 4:119–155.
- Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.
- Shepard, R. N. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2):125–140.
- Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.
- Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.
- Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419.
- Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.
- van der Maaten, L. J., Postma, E. O., and van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(1-41):66–71.
- Venna, J. and Kaski, S. (2001). Neighborhood preservation in nonlinear projection methods: An experimental study. In *Artificial Neural Networks—ICANN 2001*, pages 485–491. Springer.
- Venna, J., Peltonen, J., Nybo, K., Aidos, H., and Kaski, S. (2010). Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *The Journal of Machine Learning Research*, 11:451–490.
- Weinberger, K. Q. and Saul, L. K. (2006). An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, volume 6, pages 1683–1686.
- Wickelmaier, F. (2003). An introduction to mds. *Sound Quality Research Unit, Aalborg University, Denmark*.



# Appendix A

## Edge Bundling

This appendix is about the edge bundling technique used in Tear-up Proxigram mentioned in Chapter 5. The following discussion explains the detail of the state-of-art edge bundling method *Mingle* and my original technical contribution addressing the difficulties when it is applied on proxigram.

### A.1 Background of Edge Bundling

The edges in a graph are usually to show relationships among nodes. We found that a large number of straight-line edges can dominate the diagram, covering local details and hiding patterns. Edge bundling is an approach to reduce visual clutter. Given a fixed node layout in 2D, the technique will bundle edges that “similar” in some way into splines sharing a bundled section in the middle so that they are arranged neatly and expose more space to show other information.

There are two noteworthy methods that handle general undirected edge bundling that I would like to review in this chapter: Multilevel Agglomerative Edge Bundling [Gansner et al. \(2011\)](#) and Force-Directed Edge Bundling [Holten and Van Wijk \(2009\)](#).

Force-Directed Edge Bundling (FDEB) makes an analogy of the edge bundling problem to a physical system, where each straight-line edge is divided into segments and subjected to a linear attracting spring force to make them flexible, while another kind of electrostatic force is used between each pair of edges. In this way the edges interact, attract and morph so that edge bundling is achieved. The method is conceptually simple but computationally complex. It is claimed to produce good results but we have not found it is feasible with large number of edges. However FDEB introduces an important collection of *Edge Compatibility Measures* judging whether two edges are compatible to bundle so that they look geometrically pleasing after bundling.

To avoid the “all against all interactions” of FDEB, Multilevel Agglomerative Edge Bundling (Mingle) [Gansner et al. \(2011\)](#) considers the start point and end point of an edge as a 4-dimensional feature vector and builds a  $k$ -nearest neighbour graph in this 4D vector space which greatly reduces the number of edge-interaction computations later. Using this proximity graph, the edge bundling process is motivated by saving “ink” to draw these edges. It is an elegant and powerful solution. The next section presents Mingle in detail based on the paper [Gansner et al. \(2011\)](#), because it is the edge bundling technique that I used the research presented in chapter 7. I re-implemented Mingle in the `Python` programming language in a slightly more general way than what described in the paper and implemented the way to use the edge-compatibility measures of FDEB to improve edge bundling performance, which is mentioned in the paper [Gansner et al. \(2011\)](#).

## A.2 Edge Bundling Process

Fig. [A.1](#) is the proof of concept demonstration of how Mingle works intuitively. Suppose we are given a 2D graph  $G = \{V, E\}$ , with  $|V|$  vertices and  $|E|$  edges; Mingle algorithm has the four following steps:

1. **Edge Proximity Graph:** Build a  $k$ -Nearest Neighbour Graph  $\Gamma$  (typical value of  $k$  is small like 3 or 10) with each edge in  $G$  treated as a point in 4D space (refer to Fig. [A.1a](#) and Fig. [A.1b](#)).
2. **Agglomerative Bundling:** For each “node” in  $\Gamma$  (an “edge” in  $G$ ), try to bundle its most “ink-saving” neighbour. Some edges are bundled, the result leads some nodes in  $\Gamma$  merged and  $\Gamma$  coarsened to reflect this (refer to Fig. [A.1c](#) and Fig. [A.1d](#)).
3. **Multilevel Agglomerative Bundling:** In the last step Mingle iterates through every node in  $\Gamma$  and applies the ink-saving principle to bundle their neighbours. From then on, in each further level of agglomerative bundling like the last step, Mingle will try to bundle each node in the coarsened  $\Gamma$  with its most ink-saving neighbour whether it is a merged node or an isolated node left in the last level (refer to Fig. [A.1e](#) and Fig. [A.1f](#)). That is why it is called multilevel agglomerative bundling. The process is terminated when further ink-saving can be gained.
4. **Recursion:** sections of the bundled edges are now treated as individual edges, and the previous three steps are applied recursively, until no more ink-saving can be gained. This requires the edge proximity graph to be reconstructed based on the sections of bundled edges. (refer Fig. [A.1g](#) and Fig. [A.1h](#)).



Let any vertex in the 2D graph  $G$  be  $a$  with its 2D position as  $x_a$ . An edge  $u = (a, b)$  in  $G$  is a feature vector  $(x_a, x_b)$  which can be considered as a point in 4D space. The “ink” to draw a simple edge  $u$  is defined as  $\text{ink}(u) = \|x_a - x_b\|$ . An edge in  $G$  is considered to be “similar” to another to perform bundling, if the Euclidean distance between their 4D vectors is small. The edge proximity graph  $\Gamma$  (k-nearest neighbour graph) with its nodes representing the edges in  $G$  is constructed based on this metric, which forms a guideline for future processes (see Fig. A.1b).

With  $\Gamma$  constructed, for each neighbour  $v$  of node  $u$  in  $\Gamma$  (not to be confused with the nodes in  $G$ ), the ink-saving is calculated if the edge(s) represented by  $u$  and edge(s) represented by  $v$  are bundled. Here  $e(u)$  denotes the **set** of the edges in  $G$  represented by node  $u$  in  $\Gamma$ , since a node would represent several edges after bundling. The neighbour  $v$  with which most ink-saving is achieved is promoted to bundle with node  $u$ . The amount of ink-saving is defined as:

$$\text{ink}(e(u)) + \text{ink}(e(v)) - \text{ink}(e(u) \cup e(v)) \quad (\text{A.1})$$

To calculate the “ink” to draw for the set of edges represented by the node in  $G$  requires the following. Given a set of edges in  $G$  represented by node  $u$  in  $\Gamma$ ,  $e(u) = \{e_1 = (x_1^S, x_1^T), e_2 = (x_2^S, x_2^T), \dots, e_k = (x_k^S, x_k^T)\}$ , with their source and target points close (points in  $S$  stay close as a cluster and the same for points in  $T$ ), will result the source points set  $S = \{x_1^S, x_2^S, \dots, x_k^S\}$  and the target points set  $T = \{x_1^T, x_2^T, \dots, x_k^T\}$ , which the two sets have the same size of  $k$ . The bundling of the edges in  $e(u)$  is to have two bundle meeting points  $M_1$  and  $M_2$  where all the points in  $S$  fan-in at  $M_1$ , and then share the same bundle section  $\overline{M_1 M_2}$ , and finally fan-out at  $M_2$  to all points in  $T$ . Thus the formula of total “ink” to draw the bundle edges  $e(u)$  is

$$f(S, T, M_1, M_2) = \sum_{x \in S} \|x - M_1\| + \|M_1 - M_2\| + \sum_{x \in T} \|M_2 - x\|. \quad (\text{A.2})$$

The bundle meeting points  $M_1$  and  $M_2$  are initialised with the centroids of  $S$  and  $T$  respectively and their positions are adjusted by moving them along the line between the two centroids, at which the minimal “ink” to draw is achieved. The minimisation process requires a scalar solver such as Golden Section Search described in Press (2007).

Thus the total ink to draw  $e(u)$  bundled with  $e(v)$  becomes

$$\text{ink}(e(u) \cup e(v)) := \min_{M_1, M_2} f(S, T, M_1, M_2). \quad (\text{A.3})$$

In addition, when the source or target points are a bit faraway from their centroids and if their edges are still considered as neighbours to bundle will result in large turning angles

between the line from the point in  $S$  or  $T$  and  $\overline{M_1M_2}$ , which looks “ugly”. To avoid that a penalty scheme is used in calculating the total ink,

$$f(S, T, M_1, M_2) \left( 1 + \frac{\cos(A_{max})}{p} \right) \quad (\text{A.4})$$

where  $A_{max}$  is maximum turning angle between any point in  $S$  and the meeting point  $M_1$  or point in  $T$  and  $M_2$ . Parameter  $p > 1$  is set to tell how much penalty for large turning angle. Larger  $p$  value means ink-saving is more important than maintaining small turning angles. Typical value of  $p$  is ranging from 2 to 3. If  $p$  is too small, there might not be any bundling, since the “ink” is enlarged to the level that it is always larger than the ink before bundling.

After the *first-level* edge bundling (see Fig. A.1c), the Edge Proximity Graph  $\Gamma$  is coarsened, in that some nodes are merged into one node while some nodes remain alone if they are not bundled, and their  $k$ -nearest neighbour links are updated accordingly (see Fig. A.1d). Then the bundling process simply repeats based on the coarsened  $\Gamma$  until there is no more ink-saving could be gained (see Fig. A.1e and Fig. A.1f).

After several round of iterative agglomerative bundling, each edge will have at most three segments (or 1 segment if the edge remains unbundled with any other ones), which is the result of sharing only one bundle section. All straight-lines section of bundled edges can be seen as a set of new edges and with those unbundled ones together, the core algorithm could be applied again by building a new Proximity Edge Graph from them, which is called a recursive run (see Fig. A.1g and Fig. A.1h).

As far as I know, there is only one JavaScript implementation of this algorithm I can find in public domain [Belmonte \(2013\)](#), and the original C implementation is not available. Thus I rewrote the whole algorithm in Python from scratch and overhauled the main process in the way that Mingle can have explicit control of the number of levels of both agglomerative bundling and recursion. The Mingle algorithm pseudo-code of my implementation is given as Algorithm. 7, being slightly different from the original version in [Gansner et al. \(2011\)](#).

In the implementation of Mingle, function `coarsen( $\Gamma$ )` only occurs in multilevel agglomerative edge bundling. After this function is called, the nodes in the same group number are merged and update their neighbour links accordingly. Function `recursion( $\Gamma$ )` will do nodes merging as well but also rebuild the neighbour graph completely. To be specific, during recursion, the nodes with the same group number will be merged into a new single node like in `coarsen` but the process will view the bundled edges as individual edges. Thus like in the start of the algorithm the  $k$ -nearest neighbour graph is reconstructed based on the new collections of edges. Also only during `recursion( $\Gamma$ )`, the algorithm will insert the meeting points  $M_1$  and  $M_2$  of the bundle section shared by  $e(u)$  to the corresponding edges in  $G$ . The shape of edges changing will introduced in section. A.3.

### A.3 Edge Bundling Rendering

After the edge bundling process, each edge in  $G$  is deformed into segment(s) of bundle section(s), while the segments joint points are previous bundle meeting points  $M_1$  and  $M_2$ . The main rendering method for illustrating the bundle structure is a continuous interpolation between straight-line edges and their corresponding bundled edges, which is suggested in [Holten and Van Wijk \(2009\)](#) and adopted in [Gansner et al. \(2011\)](#). In particular, a deformed edge is denoted as a sequence of segments joint points  $\{x_0, x_1, \dots, x_i, \dots, x_k\}$ , where  $x_0$  and  $x_k$  are the source point and target point of the original edge respectively, and  $x_i$  is one of the bundle meeting points. To draw an edge in the form of spline, firstly a projection of  $x_i$  onto the straight-line  $\{x_0, x_k\}$  must be calculated,

$$\bar{x}_i = x_0 + \frac{(x_i - x_0)^T (x_k - x_0)}{\|x_k - x_0\|^2} (x_k - x_0) \quad (\text{A.5})$$

The final control point  $x'_i$  is a linear interpolation between the projection point  $\bar{x}_i$  and original segments joint  $x_i$ , written as,

$$x'_i = \lambda \bar{x}_i + (1 - \lambda) x_i \quad (\text{A.6})$$

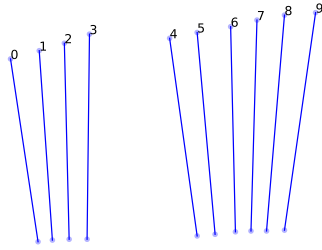
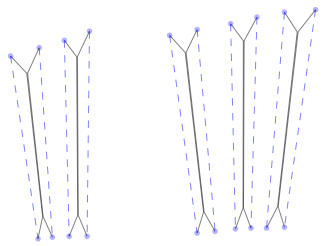
where  $\lambda$  is a control parameter ranging from 0 (final drawing point close to straight-line) to 1 (close to bundled segments). Finally with the sequence  $\{x_0, x'_1, \dots, x'_i, \dots, x_k\}$  of control points forming a spline, the edge rendering can apply, either using straight-lines or Bézier curves (See Fig. [A.2](#) for proof of concept).

### A.4 Edge Bundling Performance of Mingle

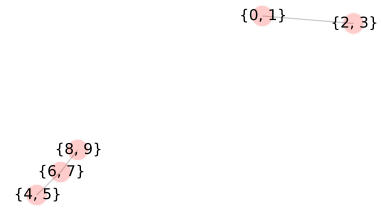
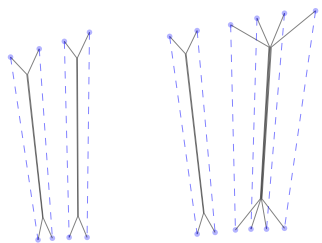
Finally to demonstrate the performance of Mingle on large edges collection, I test the algorithm on the dataset named `eastwestcommute` provided in [Belmonte \(2013\)](#), which has 130 edges massively crossing each other. This section presents a sequence of Mingle bundling edges exhibiting the effects of different parameter settings. The default parameters for Mingle are listed in Table. [A.1](#).

numOfLevels	unlimited
numOfRecursion	1
$k$	10
$p$	2
$\lambda$	1
Rendering	Straight-line

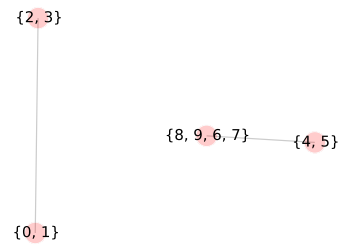
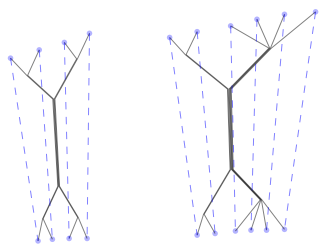
Table A.1 Default Mingle Parameter Settings

(a) Original Edges in  $G$ (b) Edge Proximity Graph  $\Gamma$  initial nodes state;  
2-nearest neighbour graph

(c) Level 1 Agglomerative Bundling

(d) Level 1  $\Gamma$  nodes state

(e) Level 2 Agglomerative Bundling

(f) Level 2  $\Gamma$  nodes state

(g) Recursive Bundling

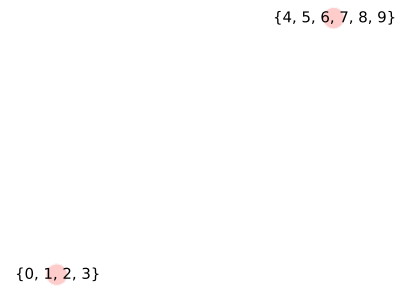
(h) Recursive  $\Gamma$  nodes state

Fig. A.1 Mingle Algorithm Proof of Concept:  $numOfLevels = 2, numOfRecursion = 1, penalty = 2, k = 2$ .

**Algorithm 7** Multilevel Agglomerative Edge Bundling Core Algorithm

**Input:** a set  $E$  of edges in  $G$ , in the form of a 4D vector  $(x_u, x_v)$ .

**Require:**  $numOfLevels, numOfRecursion$

**Init:** build Edge Proximity Graph  $\Gamma = \{N_E, E_E\}$  from  $E$ , where  $N_E$  is nodes set and  $E_E$  to be the neighbour link among them.

$totalGain \leftarrow 0$

$UNGROUPED \leftarrow -1$

**Procedure:**

**while true do**

$gain \leftarrow 0$  {current level gain}

$k \leftarrow 0$  {current group number}

$groups[u] \leftarrow UNGROUPED$ , for all  $u \in N_E$

**for each node  $u$  in  $N_E$  do**

**if  $groups[u] = UNGROUPED$  then**

            find the most ink-saving neighbour  $v$  of all neighbours of node  $u$  in  $\Gamma$

$gain_{u,v} \leftarrow ink(e(u) \cup e(v)) - (ink(e(u)) + ink(e(v)))$

**if  $gain_{u,v} > 0$  then**

                bundle  $e(u)$  and  $e(v)$

$gain \leftarrow gain + gain_{u,v}$

**if  $groups[v] \neq UNGROUPED$  then**

$groups[u] \leftarrow groups[v]$

**else**

$groups[v], groups[u] \leftarrow k$

**end if**

**else**

$groups[u] \leftarrow k$

**end if**

$k \leftarrow k + 1$

**end if**

**end for**

**if  $currentLevel < numOfLevels$  and  $gain > 0$  then**

$\Gamma \leftarrow coarsen(\Gamma)$  {nodes in  $\Gamma$  with the same group number are coalesced}

$currentLevel \leftarrow currentLevel + 1$

$totalGain \leftarrow totalGain + gain$

**else**

**if  $currentRecursion < numOfRecursion$  then**

$\Gamma \leftarrow recursion(\Gamma)$  {from merged nodes and left alone nodes in current  $\Gamma$  to rebuild  $\Gamma$ ; Meanwhile insert meeting points  $M_1$  and  $M_2$  to their corresponding edges in  $G$ , so that the edges are morphing along with edge bundling process.}

$currentLevel \leftarrow 0$

$currentRecursion \leftarrow currentRecursion + 1$

**else**

$\Gamma \leftarrow recursion(\Gamma)$

**Break**

**end if**

**end if**

**end while**

**return**  $\Gamma, totalGain$

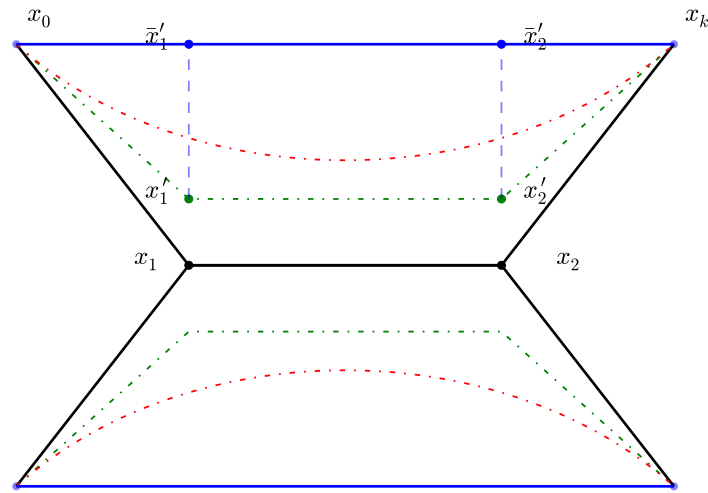


Fig. A.2 Proof of Concept of Edge Bundling Rendering: Two blue straight-line edges are bundled into a three-segments spline, shown by black lines. The Green dashed splines are the straight-line rendering of the spline with  $\lambda = 0.7$ , while the red dashed splines are the same but draw in Bézier curves.

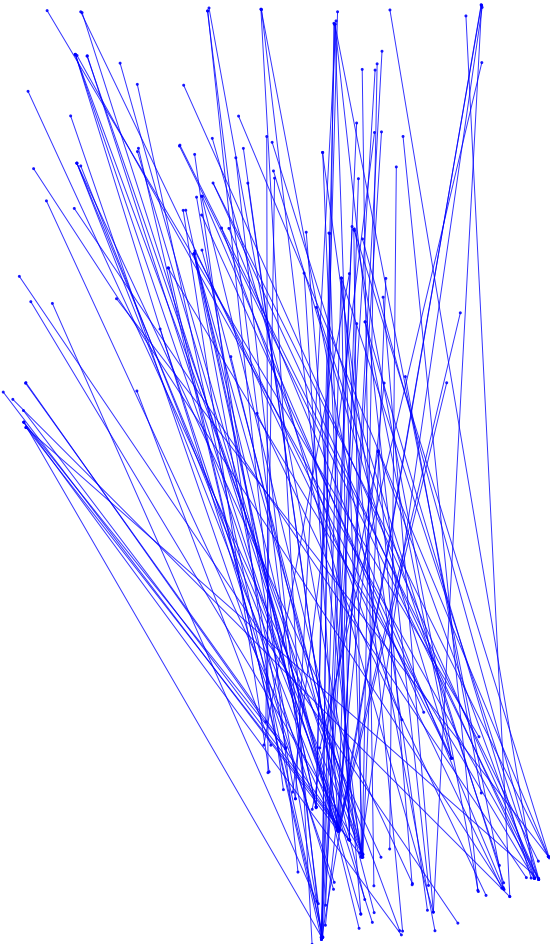


Fig. A.3 eastwestcommute dataset  
with 130 edges unbundled.



Fig. A.4 Mingle with only agglomerative edge bundling, edges reduced from 130 to 22.





Fig. A.5 Mingle after one recursive, edges reduced from 130 to 6.



Fig. A.6 The same setting as Fig. A.5, but  $\lambda = 0.7$ . The splines are rendered in straight-line. Setting  $\lambda$  not to 1.0 is not good, we should avoid in later applications because even all edges are bundled the rendering actually not saving ink but waste more ink than unbundled.

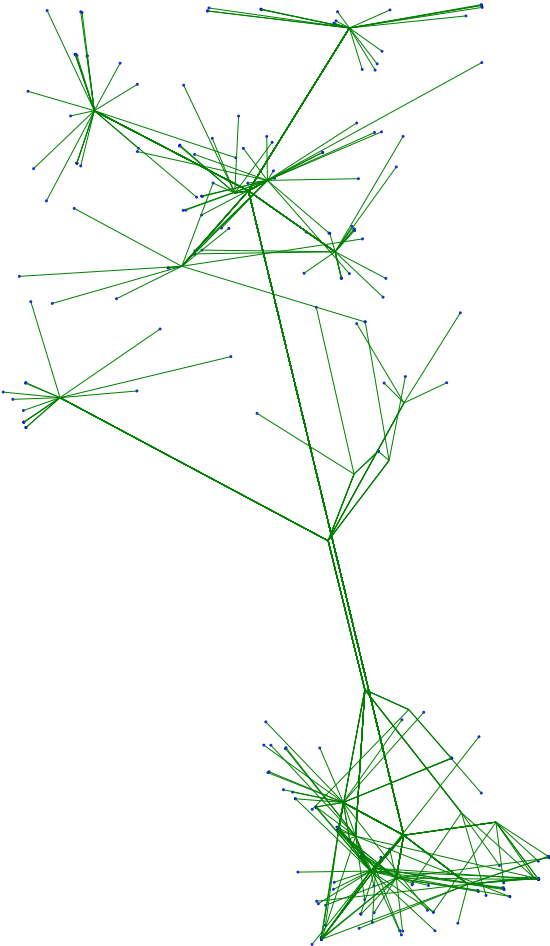


Fig. A.7 The same setting as Fig. A.5 but  $p = 5$ , edges reduced from 130 to 2. Larger  $p$  will favour bundling than small turning angle. Thus the result has more edges bundled than before, however the appearance of large turning angles are not only unaesthetic but also useless, because such aggressive bundling does the opposite of revealing high level edge patterns.



Fig. A.8 A showcase of the specific control of level of recursive and agglomerative bundling in Mingle Python implementation. Recursive times is set to 5 and Agglomerative bundling times to 1. Edges are reduced from 130 to 3. This will result the bundling have more segments than only 3, which can help the splines look more natural. In some cases specific control of coarsening levels can help avoid aggressive bundling during aggressive bundling process.



Fig. A.9 The same settings as Fig. A.8, but rendering in Bézier curves. It looks aesthetic and it is easy to follow where each edge is going. However like Fig. A.6, it does waste more ink to draw than the unbundled edges.

## A.5 FDEB Edge Compatibility Measures

One limitation of Mingle is that when building the Edge Proximity Graph  $\Gamma$ , the Euclidean distance between two 4D feature vectors is less capable of accurately describing the similarity of two edges in  $G$  than FDEB edge compatibility measures, although [Gansner et al. \(2011\)](#) argues that it is not a big issue since in agglomerative bundling, Mingle measures bundling compatibility by the most ink-saving. But there are two potential problems:

- There are always  $k$ -nearest neighbours for each edge in  $G$ , even the ones have the very least compatibility measures of FDEB, if there are no suitable edges left. Ink-saving can not always guarantee these edges are not bundled, because there might be one pair of edges have poor similarity but nevertheless ink-saving. And if it happens, the bundling looks ugly.
- When there are sufficient edges in the early stage of bundling, ink-saving usually will choose suitable edges to bundle. But after agglomerative bundling, there are few edges left, the less capability of measuring compatibility becomes significant.

As [Gansner et al. \(2011\)](#) mentions in the paper, it is possible to modify Mingle by employing FDEB edge compatibility measures [Holten and Van Wijk \(2009\)](#) for more accurate geometric consideration; As far as we are aware of, such work has not been published yet.

FDEB introduces four kinds of edge compatibility measures based on *angle*, *scale*, *position* and *visibility*, all ranging from 0 to 1. These measures are designed to address the problem that the edges are bundled too much by electrostatic forces in FDEB.

Let  $P$  and  $Q$  denotes any two edges in  $G$ . Firstly two perpendicular edges should not be bundled while if they are parallel they are good bundle candidates. Thus measure  $C_a(P, Q)$  related to the angle between two edges  $P$  and  $Q$  is introduced. The closer the angle to  $\pi/2$  is the closer  $C_a(P, Q)$  to 0, meaning the less angle compatibility two edges are. The following formula gives the definition:

$$C_a(P, Q) = |\cos(\alpha)| = \left| \frac{P \cdot Q}{|P||Q|} \right|,$$

where  $\alpha$  is the angle between  $P$  and  $Q$ .

Secondly two edges should not be bundled if their lengths differ greatly. Measure  $C_s(P, Q)$  considers the scale difference between two edges  $P$  and  $Q$ . If  $|P| = |Q|$ ,  $C_s(P, Q)$  is 1; While if  $|P| \gg |Q|$ ,  $C_s(P, Q)$  is 0.

$$C_s(P, Q) = \frac{2}{l_{avg} / \min(|P|, |Q|) + \max(|P|, |Q|) / l_{avg}},$$

$$\text{where } l_{avg} = \frac{|P| + |Q|}{2}.$$

Thirdly edges are too far away from each other should not be bundled.  $C_p(P, Q)$  is actually a distance metric for edge  $P$  and  $Q$ . If  $P$  and  $Q$  are completely overlapped,  $C_p(P, Q)$  becomes 1; The further two edges stay apart, the measure approaches to 0.

$$C_p(P, Q) = \frac{l_{avg}}{l_{avg} + \|P_m - Q_m\|},$$

where  $P_m$  and  $Q_m$  are midpoints of  $P$  and  $Q$ .

Lastly after all three considerations above, two edges could be parallel, equal-length and close to each other but have an offset, which an extreme case would be look like a skewed parallelogram. If such two edges are bundled, their bundle section would be less visible. Measure  $C_v(P, Q)$  is the minimal of two visibility measures  $V(P, Q)$  and  $V(Q, P)$ . To determine  $V(P, Q)$  for example, a projection of the edge  $Q = \overline{Q_0Q_1}$  on to the line across the edge  $P = \overline{P_0P_1}$  is made, denoted as  $\overline{I_0I_1}$ , and  $I_m$  as its midpoint. If  $P_m = I_m$ , edge  $P$  and  $Q$  coincide and  $C_v(P, Q) = 1$ , which is the ideal situation for bundling. If  $|Q|$  is projected outside  $\overline{P_0P_1}$  meaning  $\|P_m - I_m\| \geq \|Q_0 - Q_1\|/2$ ,  $V(P, Q)$  becomes 0, hence  $C_p(P, Q) = 0$ .

$$C_v(P, Q) = \min(V(P, Q), V(Q, P)),$$

$$\text{where } V(P, Q) = \max\left(1 - \frac{2\|P_m - I_m\|}{I_0 - I_1}, 0\right),$$

and  $I_m$  is midpoint of  $I_0$  and  $I_1$ .

Finally taking all four considerations, the edge compatibility measure between any two edges  $P$  and  $Q$  is  $C_e(P, Q) = C_a(P, Q) \cdot C_s(P, Q) \cdot C_p(P, Q) \cdot C_v(P, Q)$ , with its value ranging from 0 to 1. To address the potential problem in Mingle, I made two modifications in order to utilise the four edge-compatibility measures of FDEB: edge similarity metric and ink function penalty scheme. But we should be aware that the four edge-compatibility measures in FDEB are used in the electrostatic force between two edges, there will be many alternative ways of adopting these measures in modification.

### A.5.1 Edge Similarity Modification

Accurate edge “similarity” measure gives better candidates in the finite set of k-Nearest neighbours of each in  $G$ . The Euclidean distance metric of two edge vectors actually gives equal-weight for every feature of the vector in terms of the metric value contribution. Thus a similarity metric for any two edge  $P = \{P_0, P_1\}$  and  $Q = \{Q_0, Q_1\}$  like  $\|P_0 - Q_0\| + \|P_1 - Q_1\|$  would be a better choice and it is more natural if taken the midpoint of  $P_m$  and  $Q_m$  of  $P$  and  $Q$  into consideration. Also  $C_p(P, Q)$  in FDEB is an edge-position related measure, which could be seen as a similarity metric as well.

The similarity of edges to bundle has no ground truth to measure. It is hard to justify whether any metric would be better than any other, but we have to merely rely on our subjective judgement and empirical observation. The following three possible edge “similarity” metric are defined as below:

$$d(P, Q) = \|P_0 - Q_0\| + \|P_1 - Q_1\| + \|P_m - Q_m\| \quad (\text{A.7})$$

$$d(P, Q)_{C_p} = C_p(P, Q)^{-1} \quad (\text{A.8})$$

$$d(P, Q)'_{C_p} = \frac{\|P_0 - Q_0\| + \|P_1 - Q_1\| + \|P_m - Q_m\|}{C_p(P, Q)} \quad (\text{A.9})$$

Previous experiments concluded that  $d(P, Q)$  is similar to Euclidean metric,  $d(P, Q)_{C_p}$  is reliable, and  $d(P, Q)'_{C_p}$  is a better choice.

### A.5.2 Ink Function Penalty Scheme

The ink-saving principle is not reliable for selecting compatible edges to bundle. Just as Mingle uses a penalty scheme to maintain small turning angle (see Equation. A.4), I uses the remaining three edge-compatibility measure to form the penalty scheme to prevent incompatible edges to bundle.

Firstly, for any two edges  $P$  and  $Q$  in  $G$ , the combined edge-compatibility measure  $C_e(P, Q)'$  is defined as  $C_e(P, Q)' = (C_a(P, Q) + C_s(P, Q) + C_v(P, Q))/3$ . It is different from  $C_e(P, Q)$  in FDEB, simply because the three values ranging from 0 to 1 times together would be uncontrollable small for the penalty scheme. Secondly an edge-compatibility measure for a set of edges  $e(u)$  is needed, because a set of edges are the input for the ink function in Mingle. We should aware that even there is a single edge in the  $e(u)$  is incompatible with other edges in the set, the bundle will look ugly. Thus the group edge-compatibility measure is proposed to be the minimal value of  $C_e(P, Q)'$  for any two different edges  $P$  and  $Q$  in  $e(u)$ .



Thirdly the penalty scheme I defined for edge-compatibility is similar to Equation. A.4, it scales the ink value gently for group edge-compatibility measure close to 1, but punishes greatly for the measure close to 0, in which case  $\text{ink}(e(u) \cup e(v))$  will be significantly larger than  $\text{ink}(e(u)) + \text{ink}(e(v))$ . For following formulas give the definitions for group edge-compatibility measure and its ink function penalty scheme:

$$C_e(e(u) \cup e(v))' = \min_{P, Q \in e(u) \cup e(v)} C_e(P, Q)', \quad (\text{A.10})$$

$$\text{ink}(e(u) \cup e(v))' = \text{ink}(e(u) \cup e(v)) \cdot (1 - \log(C_e(e(u) \cup e(v))') \cdot s), \quad (\text{A.11})$$

$$\text{where } s \text{ is scalar, typical value is } 2. \quad (\text{A.12})$$

## A.6 Conclusion

This appendix reviews the edge bundling method Mingle and the augmentation of integrating the four edge-compatibility measures in FDEB addressing the potential issue in the algorithm. The advantage of edge bundling is that it can de-clutter a graph and allow long edges to be drawn without covering too much of the visualisation. This is a surprising benefit for a readable diagram.

Unfortunately a disadvantage is that it can be hard for the eye to track the end points of an edge if bundled. Also the bundled edges share some of the same segments, thus it is difficult to colour the edges with their own information.

Drawing separate parallel bundled segments by setting  $\lambda < 1.0$  or rendering the bundling splines in curves exposes the individual edges but consequently these two alternatives waste more “ink” than unbundled. Thus the usage of Mingle, unless specified, I only present edge bundling splines rendering in straight-lines and with  $\lambda = 1.0$  in this thesis.

All the methods of edge bundling are heuristic, using plausible ideas for procedure to bundle the edges. Unfortunately the bundling methods do not optimise any principled criteria for making edges more clearly visible, except perhaps ink-saving. To develop a principled visibility measure that could be optimised would require future research.